Development of Cooperative Behavioural Model for Autonomous Multi-Robots System Deployed to Underground Mines

Thesis by
Chika Ogochukwu Yinka-Banjo

Submitted in Fulfilment of the Requirements
for the Degree of
Doctor of Philosophy
in
Computer Science
at the
University of Cape Town, South Africa

Supervisors

Dr Antoine Bagula
Prof. Isaac O. Osunmakinde

March 2015

Development of Cooperative Behavioural Model for Autonomous Multi-Robots System Deployed to Underground Mines

by

Chika Ogochukwu Yinka-Banjo

Declaration

I declare that this thesis is my own original work. Where collaborations with other researchers are involved, or materials generated by other researchers are involved, or materials generated by other researchers are included, the parties and/or materials are acknowledged or explicitly referenced as appropriate.

This work is being submitted for the degree of Philosophy in Computer Science of the University of Cape Town, South Africa. It has not been submitted to any other university or institution for any other degree or examination.

Name: **Chika Ogochukwu Yinka-Banjo**

Signature: 

Date: 20 April 2015

Dedication

I dedicate this thesis to my husband, Adeyinka Banjo, for being there for me throughout the entire doctoral program. To my children, Victory and Ruth, your understanding and perseverance was an inspiration. You have consistently helped me keep perspective on what is important in life and shown me how to deal with reality. Finally, to the new addition to the family, Joseph Prince, your coming is a miracle and has made the whole thesis journey beautiful. You all are loved and blessed.

Acknowledgements

There are a number of people without whom this thesis might not have been written, and to whom I am greatly indebted. First and foremost I want to give all glory to God for the great things He has done and continues to do in my life, one of which is seeing to my completion of this thesis.

To my supervisors, Dr Antoine Bagula of the University of Cape Town (UCT) and Prof. Isaac Osunmakinde of the University of South Africa (UNISA) whose advice, insightful criticism and patient encouragement aided the writing of this thesis in innumerable ways. I thank them especially for their support and contributions. I would also like to thank the authorities of the University of Lagos, Nigeria for granting me a sabbatical to achieve this project, especially the staff of the department of computer science. I will not forget to say thank you to Prof. Abass and Prof. Uwadia for all your support. Thank you to the African Institute for Mathematical Sciences (AIMS) for a good foundation that led to this work.

I would like to acknowledge and thank the Organization for Women in Science for Developing Countries (OWSD) for a three year full sponsorship of this project and also special thanks to L'Oreal-Unesco for women in science sponsors for supporting and providing assistance to finish this project and even continue in the future. I must acknowledge as well the many friends, colleagues, students, teachers, and others who assisted, advised, and supported my research and writing efforts over the years. Especially, I need to express my gratitude and deep appreciation to Prof. Kingsley Fregene of the University of Pennsylvania for your mentorship, friendship and knowledge sharing throughout this work. Thanks also to Johnken Anyanwu for your numerous networking connections. Thanks so much to my church brothers and sisters. You have consistently helped me keep perspective on what is important in life and shown me how to deal with reality. I thank my parents Mr and Mrs C.Y. Ohalete and my father inlaw Mr Victor Banjo for all their daily prayers and support. Finally, to my family at all levels, for your prayers and love throughout this season and always.

Abstract

The number of disasters that occur in underground mine environments monthly all over the world cannot be ignored. Some of these disasters for instance are roof-falls; explosions, toxic gas inhalation, in-mine vehicle accidents, etc. can cause fatalities and/or disabilities. However, when such accidents happen during mining operations, rescuers find it difficult to respond to it immediately. This creates the necessity to bridge the gap between the lives of miners and the product acquired from the underground mines by using multi-robot systems.

This thesis proposes an autonomous multi-robot cooperative behavioural model that can help to guide multi-robots in pre-entry safety inspection of underground mines. A hybrid swarm intelligent model termed, QLACS, that is based on Q-Learning (QL) and the Ant Colony System (ACS) is proposed to achieve cooperative behaviour in a MRS. The intelligent model was developed by harnessing the strengths of both QL and ACS algorithms. The ACS is used to optimize the routes used for each robot while the QL algorithm is used to enhance cooperation among the autonomous robots. The communication within the QLACS model for cooperative behavioural purposes is varied. The performance of the algorithms in terms of communication was evaluated by using a simulation approach. An investigation is conducted on the evaluation/scalability of the model using the different numbers of robots. Simulation results show that the methods proposed in this thesis achieved cooperative behaviour among the robots better than state-of-the-art or other common approaches. Using time and memory consumption as performance metrics, the results reveal that the proposed model can guide two, three and up to four robots to achieve efficient cooperative inspection behaviour in underground terrains.

Table of Contents

Declaration i
Dedication ii
Acknowledgements iii
Abstract iv
Table of Contents v
List of Figures vii
List of Tables ix
List of Abbreviations xi
1. Introduction 1
1.1 Motivation 2
1.2 Problem Statements and Objectives 6
1.2.1 Research Questions 7
1.3 Contribution 9
1.4 Declaration of Recent Research 10
1.5 Organization of the Thesis 12
1.6 Chapter Summary 12
2. Literature Review and Theoretical Background 13
2.1 Robotics, Mines and Safety Fears 13
2.1.1 What is Safety? 13
2.1.2 Underground Mine Disasters 15
2.1.3 Robotics-Related Disasters 19
2.2 Cooperative Behavioural Modelling Framework 21
2.3 Deploying Cooperative Behavioural Models in MRS Navigation 26
2.4 Different Cooperative Behavioural Models 29
2.4.1 Behavioural Models Based on Collision Avoidance Success 29
2.4.2 Behavioural Models Based on Intelligence 30
2.4.3 Behavioural Models Based on Scalability 30
2.5 Open Research Issues in Behavioural Modelling for Autonomous MRS 31
2.6 Machine Learning 31
2.6.1 Partially Observable Markov Decision Processes 33
2.6.2 Reinforcement Learning 34
2.6.3 Q-Learning Algorithm 38
2.7 Swarm Intelligence 39
2.7.1 Ant Colony Algorithms 40
2.7.2 Ant Colony System 42
2.8 Chapter Summary 43
3. Proposed Behavioural-Based Model for MRS 44
3.1 Introduction 44
3.2 Proposed Cooperative MRS Behavioural Framework 44
3.3 Problem Formulation 47
3.4 Basic Navigation and Cooperative Behavioural Models 52
3.5 The Navigation Model 54
3.5.1 ACS Model Development 55
3.6 The Communication Model 57
3.6.1 QL Model Development 58
3.7 The Hybrid Model Development 59
3.7.1 Pseudocode for QLACS 62

3.7.2 QLACS Hybrid Approach Evolution 63
3.7.3 Communication and Search Cost 65
3.7.4 Illustrations 66
3.8 Chapter Summary 70
4. Experimental Evaluations of QLACS 71
4.1 Experimental Setup 71
4.2 Experiment 1: Performance of QLACS without Cooperation 73
4.3 Experiment 2: Performance of QLACS with Good Cooperation 74
4.4 Experiment 3: Performance of QLACS for the Navigation Behaviour of the Proposed Model 75
4.5 Experiment 4: Benchmarking the New Hybrid Model (QLACS) with Popular Methods 76
4.6 Experiment 4: Benchmarking the New Hybrid Model (QLACS) with Popular Methods 77
4.7 Chapter Summary 79
5. Scalability Analysis of QLACS 80
5.1 Implementation Design 80
5.2 Experiment 1: Performance of QLACS Using Two Robots 83
5.3 Experiment 2: Performance of QLACS Using Three Robots 84
5.4 Experiment 3: Performance of QLACS Using Four Robots 86
5.5 Experiment 4: Observations of Time and Memory Scalability 87
5.6 Comparative Analysis of Cooperative Behaviour Systems 89
5.7 Discussions and Findings 90
5.8 Chapter Summary 91
6. Conclusion and Future Work 92
6.1 Summary and Conclusion 92
6.2 Recommendation 93
6.3 Future Work 93
Appendix A 95
List of References 97

List of Figures

1.1 A list of robot's applications and capabilities for recent research. The applications and capabilities highlighted in blue are the specific focus of this thesis……..1

1.2 The linkages of mining to South African economy……..……..…3

1.3 SA's mineral value: 6 main commodities…….……..4

1.4 Total world underground ore production by continent in Meitnerium……...…....5

1.5 Overview of multi-robots supporting humans and performing cooperative inspection in an underground mine……..…….....7

2.1 Safety definitions……..…….....14

2.2 Fatality per casualty classification……..……..……...…...15

2.3 Injury per casualty classification……..……..……..……...15

2.4 The four phases of a disaster…….....18

2.5 Types of breakdown that affect robots' reliability and safety……..…...…...19

2.6 Do existing behavioural models need improvement for robotics and/or mine safety?.....21

2.7 Robot navigation problems……...…...……..….27

2.8a MRS open research issues on cooperative behavioural modelling…..……..….31

2.8b MRS research topics and their key open issues……...……..…..31

2.9 Learning system model…….....32

2.10 Are machine-learning systems safe enough to avoid collisions?.....33

2.11 Reinforcement learning cycle with cause and effect……..……...……...…35

2.12 Reinforcement learning in robotics…….…....36

2.13 List of reinforcement learning…….……..…..37

2.14 The taxonomy of stochastic algorithms……..…...…….……40

3.1 Framework of the proposed QLACS model for cooperative behaviours……...……...45

3.2 Breakdown of the framework. (a) contributions of the framework, layer by layer and (b) processes of the multi-robot behavioural system……..……..……..……46

3.3 Does scalability contribute to safety?.....47

3.4 Model of the environment with two entrances and exits (2EE)……...…...48

3.5 Events and transition diagram of the modelled environment with H as goal state……49

3.6 Weighted map/graph of the model environment……...……...…54

3.7 Pseudocode of the route-finding using ACS algorithm……...……....56

3.8 Pseudocode of the behavioural model using Q-Learning algorithm…...……..…..58

3.9 Pseudocode for QLACS……...……..…..63

3.10 Hybrid architecture....……...……..…63

3.11 Parameters for QLACS example I..…..……..66

3.12 QLACS example I navigational analytical solution…….……..…..67

3.13 Parameters for QLACS example II……...…….……68

3.14 QLACS example II cooperative behaviour...……...…..…..70

4.1 Different experimental behaviours for two robots…….....……72

4.2 Processes used in achieving Tables 5.1 and 5.2…….....…...74

4.3 Comparison of time costs for QL, ACS and QLACS……...…...…..…77

4.4 Comparison of route costs for QL and QLACS...…….....……...78

4.5 Number of ants/iterations required for QLACS to find optimal route……...79

4.6 Time and iterations required for QLACS to achieve optimal behaviour……...….79

5.1 Underground terrain with segmented regions and two robots……..80

5.2 Class diagrams showing interactions of objects……...……..….82

5.3 Movement of two robots showing cooperation……...…..…..84

5.4 Movement of three robots showing cooperation…...……..…86

5.5 Movement of four robots showing cooperation……...…...87

5.6 Trends of time with number of robots……...……..…88

5.7 Trends of memory usage with number of robots……...……...…..88

List of Tables

1.1 Some specific disaster reports in South African underground mines ……..……5

2.1 Coal mine underground fatality for South Africa and other countries…..……16

2.2 Disasters in underground mines……..16

2.3 Q-learning algorithm……..……38

2.4 Variant of ACO……41

3.1 State and possible actions of the environment……...49

3.2 Initial reward matrix……..50

3.3 Combined adjacency and weight matrix……..……..54

3.4 Methods and functions for the ACO framework…….……...55

3.5 Methods and functions for the cooperative behaviour framework…….……...57

3.6 A list of parameters for the ACS model…..……..59

3.7 A list of parameters for the QL model……...59

3.8 Pheromone update of a full cycle…..……..……67

3.9 Probability update of a full cycle……..……68

4.1 QLACS without communication (inspecting some states).....……73

4.2 QLACS without communication (inspecting all states).....……..73

4.3 QLACS with communication…..……..……74

4.4 Navigation behaviour parameter specification……76

4.5 Navigation behaviour computations…..……...76

4.6 New hybrid model QLACS computations……..77

4.7 Time cost comparison for QL, ACS and QLACS…..……..77

4.8 Route cost comparison for QL and QLACS……78

5.1 Performance of QLACS with two robots-based MRS……84

5.2 Performance of QLACS with three Robots-Based MRS……85

5.3 Performance of QLACS with four robots-based MRS……...87

5.4 Summary of scalability performance on QLACS……...89

5.5 Comparative evaluation of existing cooperative behavioural system with our proposed system……..…..90

List of Abbreviations

ACO	Ant Colony Optimization
ACS	Ant Colony System
AI	Artificial Intelligence
AS	Ant System
BBS	Behavioural Based Systems
BRL	Bayesian-descrimination-function-based Reinforcement Learning
EC	Evolution Computation
FSM	Finite State Machine
GDP	Gross Domestic Product
GPS	Global Positioning System
Hc	Hazardous Conditions
IEEE	Institute of Electrical and Electronics Engineers
LLP	Lower Left Part
LMP	Lower Middle Part
LRP	Lower Right Part
MCP	Middle Central Part
MDP	Markov Decision Process
ML	Machine Learning
MLP	Middle Left Part
MLP	Multi-layer Machine
MMAS	Max-min Ant System
MRS	Multi-robot System
POMDP	Partially Observable Markov Decision Process
QL	Q-Learning
QLACS	Q-Learning and Ant Colony System
R1	Robot 1
R2	Robot 2

R3	Robot 3
R4	Robot 4
RC	Roof Cracks
RL	Reinforcement Learning
SI	Swarm Intelligence
SVM	Support Vector Machine
TG	Toxic Gases
ULP	Upper Left Part
URP	Upper Right Part

1. Introduction

Researchers began investigating issues in multiple robot systems in the late 1980s when the field of distributed robotics originated [1]. Prior to this time, research has not involved robotic components, rather concentration is on single-robot system. Parker [1], identified eight primary research topics in multi-robot systems in 2003, which are still being studied in today's research. These include (1) biological inspirations, (2) communication, (3) architectures, task planning and control, (4) localization, mapping and exploration (5) object transport and manipulation (6) motion coordination (7) reconfigurable robotics and (8) learning.

In the past, robots have been restricted to industry; however, as research and technology advanced, exploration of other areas of interest concerning robots also evolved. These areas include exploration of an unknown planet [2], capturing intruders [3], pushing objects [4,5], selecting action in a robot soccer game [6], cooperative construction and transportation [7], cleaning up toxic waste [8], decision-making [9], joint transportation [10]. A multi-robot system (MRS) can be used to accomplish tasks that are difficult to achieve for an individual robot [11]. Figure 1.1 displays a list of application areas and robots capabilities that attract much attention to the research community of robotics.

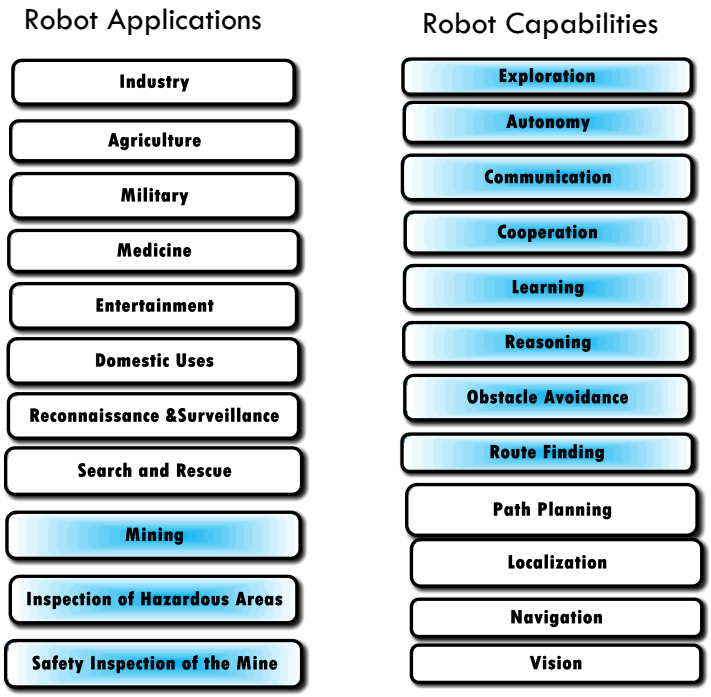


Figure 1.1: A list of robot's applications and capabilities for research. The applications and capabilities highlighted in blue are the specific focus of this thesis.

There are several benefits in using MRS to achieve tasks rather than a single robot, especially in the presence of uncertainties and incomplete information. Some of the benefits are avoiding wastage or being more economical, being able to expand to cope with increased use or being scalable. MRS also means robustness because the failure of one or more robots is not fatal to the overall system [12]. These features have made MRS an excellent choice for space application problems such as our research problem where multiple-robot inspection is faster than single-robot inspection.

For MRS to autonomously explore any environment, it requires navigation to achieve a task without colliding with obstacles and without continuous human guidance. It also requires keeping track of one another's position in the environment and most importantly communication with one another to establish cooperation and coordination that will enhance their performance.

Multi-robot learning in general and cooperative task-learning in particular, are some of the areas in which significant research into MRS remains to be done. Although an extensive amount of research work has been done for multi-agent learning, applications such as foraging, multi-robot soccer, or cooperative target observation for an MRS has not been explored extensively [1]. It will be virtually impossible for a robotic system to become a fully autonomous system without learning capabilities. Machine-learning technologies have become of crucial importance. In view of this, our research is looking at the unfinished work in multi-robot learning, i.e. applying cooperative behaviour for an MRS to an underground terrain.

1.1 Motivation

Mining industries are a significant economic sector for many countries, including the Republic of South Africa, and they incorporate the use of coal, metal and non-metal minerals. The usefulness of the minerals extracted from mining cannot be overemphasised: they are used in electrical power generation, production of steel, commercial and residential building products and asphalt, and in medicine, household, electronic and other manufactured products [13].

Most of the income of the mining sector from South Africa is spent locally. Figure 1.2[1], displays the mining contribution to the South African economy in 2012. Mining is a substantial contributor to the rapid growth of the economy, job creation, gross domestic product (GDP), fixed capital formation, gross investment and merchandise exports on global scale. The contribution of mining to South Africa over the decade prior to 2012 expressed in real money terms[25] is R2,943.27 billion sales revenue, R2160.73 billion export earnings, R2104.82 billion GDP, R705.50 billion employee remuneration and R547.81billion fixed investment.

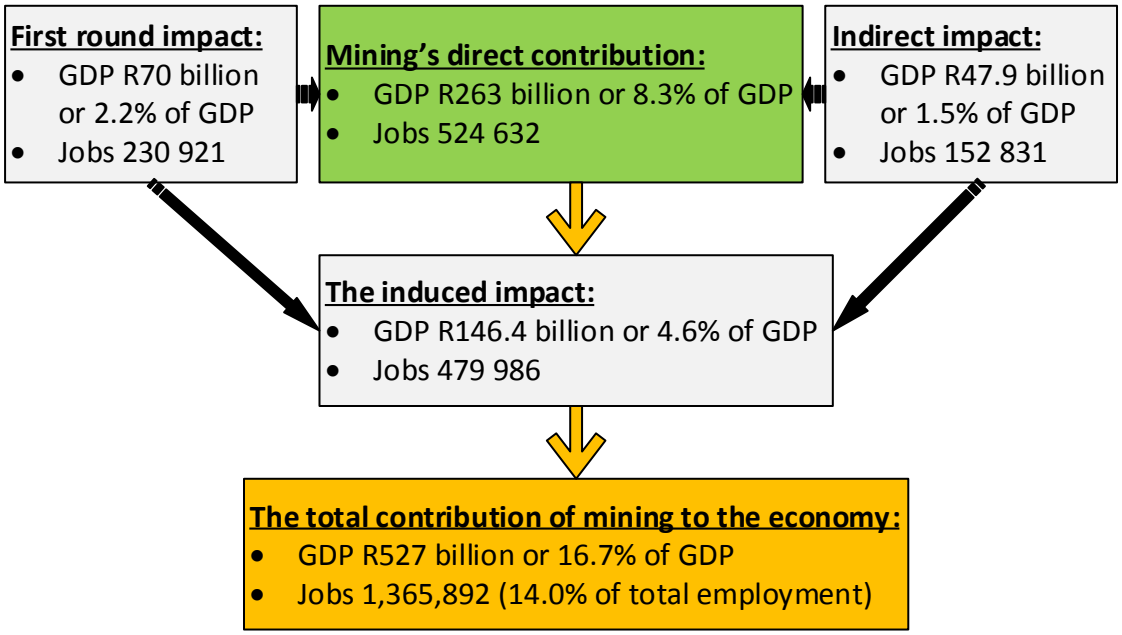


Figure 1.2: The linkages of mining to South African economy

Ore deposits are naturally occurring minerals often buried inside the earth. Some of these minerals can be extracted through surface/open-cast mining or sub-surface/underground mining [14]. Out of more than ten minerals mined in South Africa, more than half are obtained through underground mining. Figure 1.3 displays some underground mined minerals in South Africa, their mineral value in 2012 and their contribution to the nation's economy [26].

[1] Chamber of Mines of South Africa. Fact about South Africa Mining. Available online: http://www.bullion.org.za/documents/Chamber%20of%20Mines%20fact%20sheet%20August%202013_revised.pdf (accessed on 13 April, 2014).

Figure 1.3: SA's mineral value: 6 main commodities

At the same time, the tedious and dangerous nature of underground mining makes it vulnerable to mine accidents and disasters, such as the fall of loose rocks, inhalation of dangerous toxic gases (TG) beyond the human limit, trapping of miners and miners' death. The death toll in underground mining is caused by a number of factors, such as confined work space, exposure to harmful substances. Generally, these confined spaces are not designed for people to spend a lot of time in, as miners often do. These spaces are made more hazardous by poor ventilation, poor visibility and high air concentrations of combustible or toxic dust and gases, such as carbon dioxide and methane [15]. The temperature in underground mines can be relatively high, as high as 30°c and above, and the environment can be very humid, with relative humidity up to 90% and above [15]. Despite significant reduction through safety research measures, the number of disasters in underground mines in South Africa remains high.

The chart in Figure 1.4 shows that Africa and Europe are the richest in the world in terms of total underground ore production in Meitnerium compared to other continents. South Africa is the largest producer of minerals in Africa and the world's biggest underground ore producer[2]. Thus, it is not surprising that it has the highest underground fatality rate when compared to her foreign counterparts, such as the USA, Germany, India, and others [33].

[2]MiningTrends:http://www.rmg.se/RMG2005/pages/attachments/Mining_trends_atlas_Copco_200309._Trends_in_Underground_Mining.pdf (Accessed on 02 September 2012)

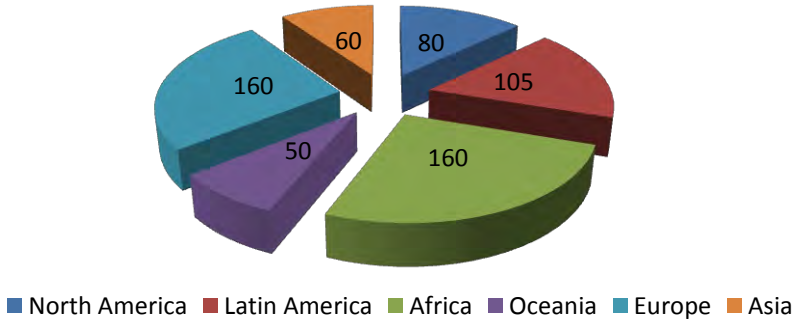


Figure 1.4: Total world underground ore production by continent in Meitnerium

Underground mining remains one of the most hazardous occupations in the world because of its disaster rate. Over the past five decades, South Africa has recorded a high number of deaths in underground mining accidents. The world's fifth worst underground mining disaster, in which 437 miners died, happened in South Africa at Coalbrook [28] mine in 1960. Several disasters have happened since then. Between 1983 and 1987, 3000 miners died and 5000 sustained permanent disabilities [16] in South African mines. In recent times, there have been a large number of disasters in South African underground mines and this is a major motivation for our research work. Table 1 displays a selection of underground mine disasters in South Africa since 2000, their causes and the number of fatalities and injuries.

Table 1.1: Some specific disaster reports in South African underground mines

Year	**Mine**	**Causes**	**Fatalities**
10 January 2000	Africa Rainbow Minerals	Fall of ground	6
15 May 2000	Beatrix	Methane explosion	7
22 Sept. 2000	Kloof Gold	Hanging rock	4
8 May 2001	Beatrix Mine	Methane explosion	13
14 Dec. 2001	Deelkraal Gold	Fall of ground	6
22 January 2002	Driefontein Gold	Fall of ground	5
17 May 2002	Great Noligwa	Fall of ground	6
17 October 2002	Argo Diamonds	Conveyance	4
30 January 2003	Driefontein Gold	Fire	5
1 April 2003	Tauton Gold	Fall of ground	5
26 May 2003	Tauton Gold	Fall of ground	4
20 Sept. 2004	Northam Platinum	Fire as a result of friction	9
13 Dec. 2004	Hernic Ferrochrome	Mud rush	7
6 July 2010	Marikana Platinum	Rock fall accident	6
8 May 2011	Beatrix Gold	Methane gas explosion	14

The main motivation of this research is to contribute to safety and the zero fatality rate milestone set by the Chamber of Mines of South Africa for her mining industry. This we intend to achieve by building a cooperative model to guide an MRS to perform a pre-entry safety inspection of the underground mine before mining operations begin. This we believe will help to give early warning to miners if the environment is not safe for operations thus reducing the fatality and disaster rates that happen in underground terrains.

1.2 Problem Statements and Objectives

Historically, underground miners received training on how to operate in underground mines by observing the rules and regulations issued by the mine health and safety organizations. However, there are uncertainties in underground mines that can cause disasters even though miners obey the rules and regulations issued. These uncertainties are rock falls, roof cave-ins, methane explosions, etc.

In this context it is worth noting that after daily mining activities in the mines, miners currently do not know the state of rocks, whether very weak, partially weak, or stable for the next mining activities and are consequently ignorant of the potential for rock falls. Similarly, old and abandoned mines are very dangerous to anyone who attempts to explore them without proper knowledge and safety training[3]. These mines often contain deadly gases, standing water that may hide deep pits, weak rocks and cracked roofs. The most important question in this context is how miners can be aware of the situation in mines before they engage in operations. Can miners know the state of rock formations, whether very weak, partially weak, or stable for mining activities and consequently gauge the potential of rock falls? The same goes for the level of toxic gases (TG) in the mine. Can miners know the level of concentration of TG in the mine before operation? Understanding these concerns through robotic automation of multi-robot cooperative behaviour will help to prevent disasters in underground mines.

[3] Mining: http://en.wikipedia.org/wiki/Mining (accessed on 04 April 2014).

This is where the environmental inspection will entail a cooperative MRS approach. The aim is to have a general standard/model for underground terrain inspection using MRS. Communication, task allocation and information access are taken into consideration in designing this cooperative behavioural system. In an attempt to proffer solutions to safety-related issues in underground mines, an MRS could play a major role. An MRS can be used to check the status of the roof and the level of TG in the mine. However, a cooperative behavioural model is required for thorough, efficient and effective inspection of underground mine safety. The outcome of this research, as depicted in Figure 1.5, will contribute a great deal to safety in underground mines and thus increase their productivity.

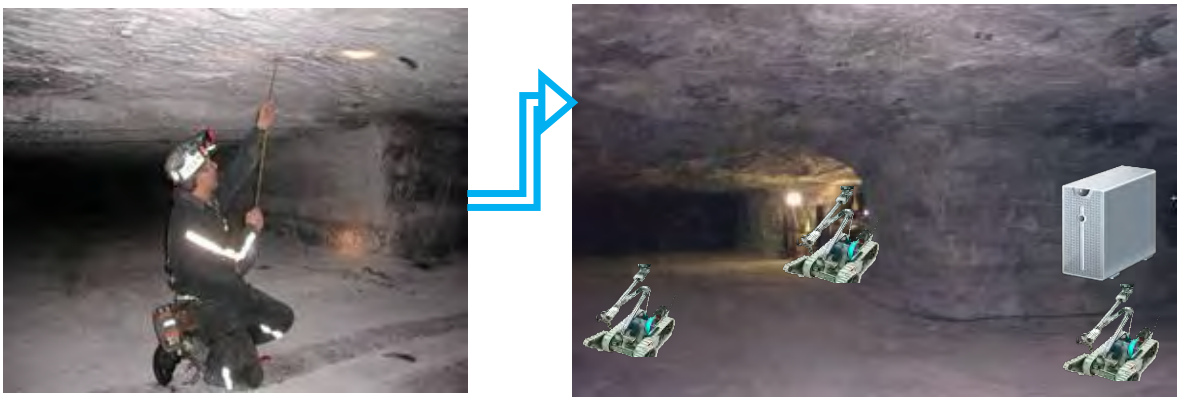


Figure 1.5: Overview of multi-robots supporting humans and performing cooperative inspection in an underground mine

1.2.1 Research Questions

Let R be a single robot. Also let $\{r_i\}$ be a set of robots.

Some tasks can be performed faster with a collective or MRS $\{r_i\}$ rather than with a single robot *R*. A typical example in this case is the issue of speed [17]. To obtain reasonable speed, the work performed by each robot must be well coordinated and each element of $\{r_i\}$ must have abilities near those of *R*. If coordination is not taken into consideration, there will be a loss of speed, as multiple robots will search the same area. Inter-related communication between the robots is also urgently required. Reliability is another issue investigated in this research. Reliability is the performance measure for which MRS easily exhibit performance over that of a single robot. This

means that failure of a single element of $\{r_i\}$ may not result in task failure, but failure of *R* guarantees task failure. System of robots is chosen for this research over independent robots because they are partially independent, self-aware and are used to solve difficult problems [102]. One major objective of MRS is to distribute both the actions/sensing of the robots and the intelligence. The following are the research questions for the development of a cooperative behavioural model by autonomous MRS in an underground terrain.

1. **Research question one**

   How can a system model of cooperative behaviour be effectively developed for autonomous MRS using machine-learning algorithms? That is, how can one develop a non-conflicting model of an MRS in a learning/swarm intelligence framework?

   A number of publications have shown how to use an MRS in a variety of tasks since four decades ago. More attention has been directed to the coordination of MRS, because a team of robots can accomplish more than can be accomplished by a single robot [18, 19]. Most of the studies done in cooperation with MRS include localization, task allocation, collision avoidance etc. [20, 21]. The motivation for this research is to enhance the security of the environment and lives of the workers in underground terrains. This means exploring effective and efficient ways of building a cooperative behavioural model that can guide MRS to accomplish pre-entry safety inspections in the underground mine before miners start operations. The inspection of the status of the mine roof and the level of TG in the mine are two specific criteria of inspection handled in this research. The resultant cooperative model achieved from Q-Learning [QL] and Ant Colony System [ACS] has not been implemented together to achieve this goal. It has not also been implemented in an underground mine.

2. **Research question two**

   How many robots in a team can be coordinated successfully before performance diminishes? What would be the scalability of the system model?

Scalability is a performance scaling parameter that investigates how the designed model handles increasing number of robots. We intend to determine the degree of scalability of the cooperative behavioural model for MRS developed in question one and establish an upper bound on the number of robots that can be effectively coordinated, before performance degradation sets in. This will determine the number of robots that can work as a team to achieve the underground terrain inspection task. Using multiple robots to assist in dangerous real-life scenarios such as in mines, security patrolling jobs and rescue operations helps in saving human lives [22]. It is better for humans to monitor tasks done in hazardous environments from a safe location as robots get the work done. Scaling our model with a number of robots will enhance productivity and safety [23].

1.3 Contribution

It is essential to address the issues of mine safety by developing an MRS cooperative behavioural model for pre-entry safety inspection of the underground mine with a hybrid artificial intelligence method. Within the MRS field of study, this thesis focuses on the learning problem that tries to answer the question: What is the behaviour of robot 1 (R1), knowing what the behaviour of robot 2 (R2) is? The following highlights the major contributions of this thesis.

We systematically survey and generate knowledge as a reference guide to understand research challenges related to mine safety, which reveals possible research interests in this area. Our illustrations of safety, robotic-related disasters and underground mine disasters show that our new model is suitable to handle the specific identified issues. Open research issues in behavioural modelling for autonomous MRS would assist in providing more problem-solving models for researchers.

We present the principles of swarm intelligence and QL algorithms. The modified version of these algorithms form the basis of the new hybrid model developed for this thesis. The mathematical

analysis reveals how the two algorithms were cleverly hybridized to form the new QLACS model. The model framework explains the three layers involved in creating this model. We describe the navigation part of the model with swarm intelligence-based technique and the communication part of the model with the QL technique. These two techniques assisted in creating the model for addressing cooperative behaviours in MRS for a safety inspection in underground terrain.

We created an underground terrain scenario that was used to explain the situation and perform some experiments on it. The description of worked scenarios will ease implementation for systems engineers, robotics researchers and practitioners. We present detailed experimental evaluations of the proposed QLACS model conducted on a simulated underground tunnel. Also, the proposed model is benchmarked with related methods. Experimental demonstration of detailed evaluation of the developed hybrid QLACS using two, three and four robots for the purpose of addressing cooperative behaviours in MRS and achieving underground terrain safety inspections was also achieved. The experimental observation of time and memory scalability on the hybrid model, and detailed comparative analysis on cooperative behaviour systems contributed to concluding our findings.

1.4 Declaration of Recent Research

The research work has yielded the following recently published articles:

Referred Book Chapter

- Osunmakinde, I.O; **Yinka-Banjo, C.O**; Bagula, A. (2012) Investigating the use of Bayesian Network and K-NN Models to Develop Behaviours for Autonomous Robots, In Raol J.R. et al. (Eds.): Mobile Autonomous Systems (2012) CRC Press. Taylor & Francis Group USA, pp 751 – 764, ISBN: 9781439863008

Referred Journal Publications

- **Chika Yinka-Banjo**, Isaac O. Osunmakinde, and Antoine Bagula, (2014) Cooperative Behaviours with Swarm Intelligence in Multirobot Systems for Safety Inspections in Underground Terrains, Journal of Mathematical Problems in Engineering, special issue in Computational Intelligence Approaches to Robotics, Automation, and Control, ISSN: 1024-123X (ISI journal) (2014), 20 pages, View/download link: ISSN: 1024-123X

- **Chika Yinka-Banjo**, Isaac O. Osunmakinde, and Antoine Bagula, (2014) Performance Evaluation of Cooperative Behaviour Model on Multi-Robot Systems for Mine Safety Inspections, International Journal of Advanced Robotics System, (ISI journal) (In Press)

- **Yinka-Banjo, C.O.;** Osunmakinde, I.O.; Bagula, A. (2012), Autonomous Multi-Robot Behaviours for Safety Inspection under the Constraints of Underground Mine Terrains, Ubiquitous Computing and Communication Journal; ISSN 1992-8424, (7) 5, pp. 1316 – 1328

- **Chika O. Yinka-Banjo**, Isaac O. Osunmakinde, and Antoine Bagula (2012) Collision Avoidance in Unstructured Environments for Autonomous Robots: A Behavioural Modelling Approach, International Journal of Advanced Materials Research, MEMS NANO and Smart Systems, (Scopus Indexed), Vol. 403 - 408,Pages 3559-3569. View/download link: ISSN: 1662-8985

Referred Conference Publications

- **Chika O. Yinka-Banjo**, Isaac O. Osunmakinde, and Antoine Bagula (2014) Multi-robot Systems: A Cooperative Behaviour and Performance Evaluations for Mine Safety Inspections. In proceedings of the South African Institute for Computer Scientists and Information Technologists (SAICSIT 2014) Conference, Pretoria, South Africa (Paper Accepted)

1.5 Organization of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 provides a discussion of literature review and theoretical background of the problem. In chapter 3, the exact nature of the problem is defined. The new hybrid model is developed; the algorithm and analytical approach are explained. The main contribution of this thesis is also explained in this chapter and the details of two new improved techniques are given.

The performance of QLACS in simulation is demonstrated in chapter 4, while the implementation of QLACS on an increasing number of robot simulations is described in chapter 5. Comparative studies of findings from the performance of QLACS with other models are carried out in this chapter. Chapter 6 presents the conclusion by summarizing the whole thesis and discusses suggested future research problems in relation to our model.

1.6 Chapter Summary

The discussion in this chapter offers a general overview of the research work. The several benefits of using MRS to achieve tasks rather than single robot were presented. The motivation for this work was explicitly stated. The research problems and objectives were also stated. How to handle the research problems was stated in the research questions and answers. The contributions of this thesis in terms of publications were listed too.

2. Literature Review and Theoretical Background

The literature review is divided into three parts. The first part deals with the background to the study contained in this thesis. The second part deals with the existing cooperative behavioural models and their paradigm, while the third part examines the underlying principles behind the intelligence of MRS models.

2.1 Robotics, Mines and Safety Fears

Roboticists have illustrated for many years the potential advantages of robots over humans. Robots can be built into different places and their sizes can range from miniature to large. Robots are robust, in that they can be designed to cope with adverse conditions such as heat, toxic chemicals and radiation, and can operate in environments that are dangerous for humans. Robots can be made to be intelligent; and are thus capable of making autonomous decisions.

Different safety fears in underground mines are categorized into mine-related and robotic-related disasters. Robotic-related disasters are classified according to random component failures, human errors, software failures and systematic hardware faults. We present an empirical analysis of how our proposed model can be used or implemented through two real-life scenarios: a) the inspection of rock falls and b) the detection of gas levels in underground mines. This thesis can be used as a reference guide to understand safety management and cooperative behavioural models for underground mines and to conduct further research on the existing models to make them more efficient, reliable and scalable, which can promote their use in larger mines and applications

2.1.1 What is Safety?

Before we proceed to findings, it is important to mention that the information technology, robotics[4] and mining disciplines have different perceptions of safety. However, the definitions of the term

[4] https://www.osha.gov/SLTC/robotics/

safety that are provided broadly underpin its dictionary[5] meaning (see Figure 2.1). From the point of view of cooperative behavioural models, the definition of safety appears to be closer to those used in the robotics and mining sphere. In robotics, self-driving cars are likened to a safety system by Seth Teller[6]. Teller claims that the number of accidents will decrease substantially with self-driving cars. In the area of smarter transportation systems, Jessica Richeri [31] also developed an autonomous vehicle able to recognize and avoid obstacles, thus, it is hoped that accidents will be reduced. Collision avoidance is another related safety area that enables robots to navigate freely to their destinations without colliding with any obstacle.

The mining[7] definition of safety includes all mining activities ranging from slips and falls to protection from electric shock, machinery, eye injury and drowning. Safety in mining involves training of miners, which helps in maintaining the relationship between working safely and productively. Figure 2.1 shows different safety definitions [65].

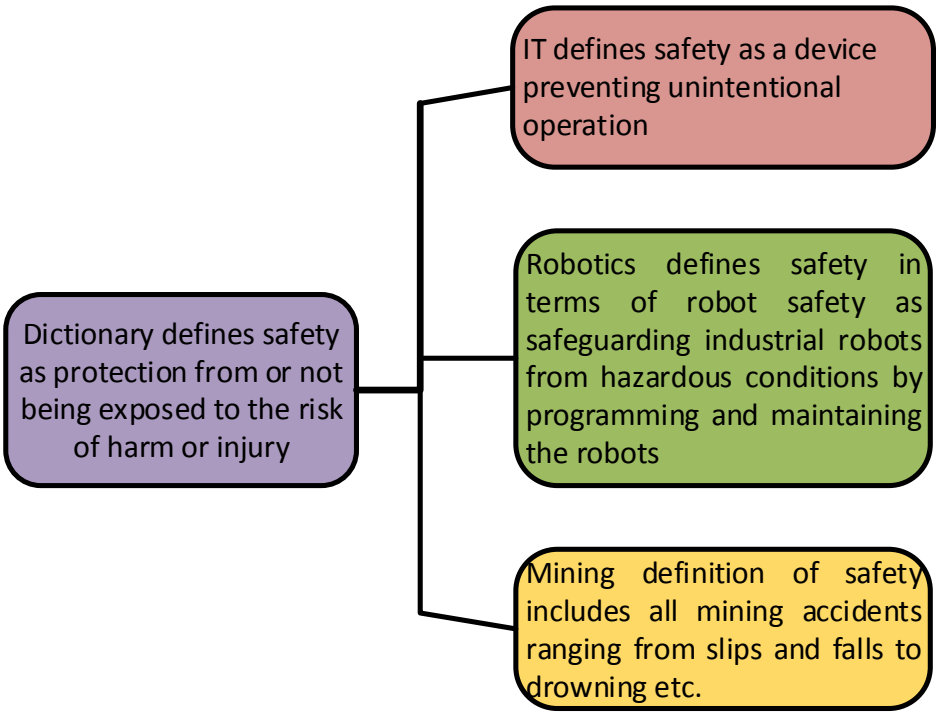


Figure 2.1: Safety definitions [65]

[5] http://dictionary.reference.com/browse/safety?s=t

[6] Popular Science: http://www.popsci.com/cars/article/2012-04/who-blame-when-robotic-car-crashes (accessed on 04 April 2014).

[7] http://www.dmr.gov.za/mine-health-a-safety.html

The researcher's views and acceptance of a cooperative behavioural model will be better understood as a safety paradigm towards the end of the thesis.

2.1.2 Underground Mine Disasters

Mine disasters are caused by explosions (methane, coal dust or others), fires, rock and roof falls, TG outbursts and inundations/rushing water [32]. Figures 2.2 and 2.3 show the recent causes of fatalities and injuries in South African mines. At present, fall of ground accidents occur mainly in deep underground mines, as shown in Figures 2.2 and 2.3.

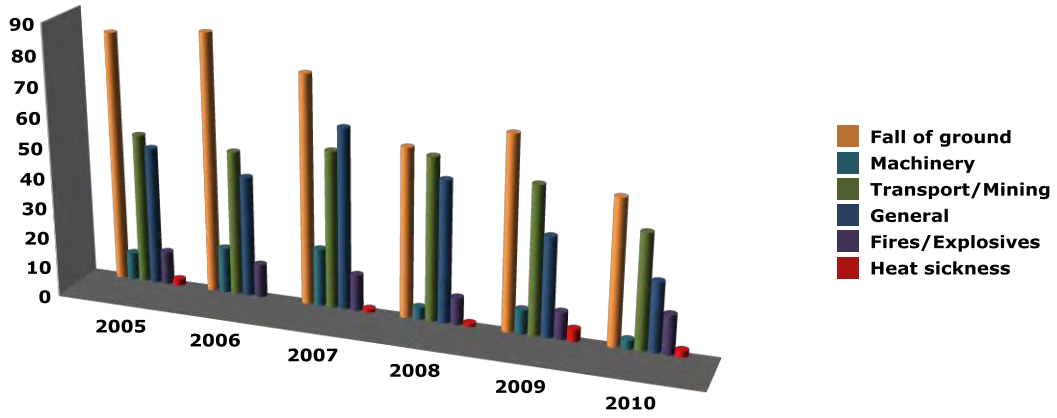


Figure 2.2: Fatality per casualty classification

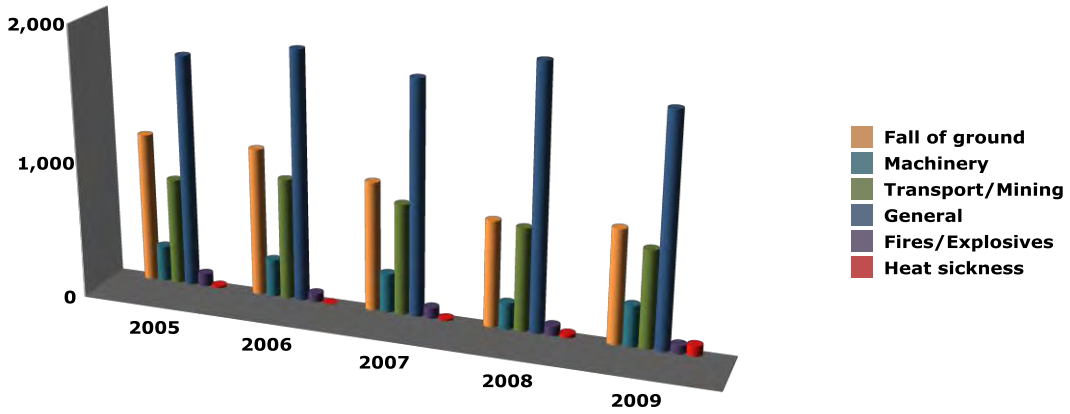


Figure 2.3: Injury per casualty classification

Table 2.1 shows a comparison between the South African coal mine fatality rate and that of other countries between year 1990 and 2004. Death rate on Table 2.1 is measured by the number of

miners killed for every million metric tons of coal produced. According to the table, the average South African fatality rate is over two times that of the Indian, five times that of USA [41] and approximately double the rate in the German.

Table 2.1: Coal mine underground fatality for South Africa and other countries [24]

Year/ Country	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004
South[8] Africa	0.53	0.48	0.65	1.57	0.96	0.53	0.75	0.72	0.73	0.51	0.54	0.38	0.44	0.45	0.53
Ger many	0.54	0.63	0.69	1.18	0.75	0.54	0.48	0.19	0.19						
India	0.28	0.24	0.23	0.25				0.54	0.47	0.45	0.46	0.45	0.29	0.33	0.27
USA	0.07	0.07	0.06	0.06	0.05	0.05	0.04	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03

In Table 2.2, some underground mine disasters from various countries and the causes are explained. The most frequent cause of these disasters was rock fall. The fatality numbers in Table 2.2 show that South Africa had the highest number (435), followed by India (375).

Table 2.2: Disasters in underground mines [32]

Year	Mine	Causes	Fatalities
6 August 2007	Crandall Canyon Mine, USA	A roof collapse trapped 15 miners in an underground coal mine	9 deaths 6 injuries
25 April 2006	Beaconsfield Tasmania Mine, Australia	A small earthquake triggered an underground rock fall, trapping 17 miners	1 death
27 December 1975	Chasnala Mine, India	An explosion that was followed by flooding and then roof cave-in drowned 375 miners trapped underground.	375 deaths
21 January 1960	Coalbrook North Mine South Africa	This disaster was caused by the collapse of several walls and a series of rock falls, releasing large quantities of methane.	435 deaths
5h August 2010	San Jose Cooper-gold Mine, Chile	A roof cave-in trapped 33 miners in a chamber of about 2300 feet below the surface.	0

Mining companies and researchers have been pursuing the goal of mitigating some of these mine disasters. A holistic five-point [37] ground management strategy to address the incidence of accidents and fatalities related to fall of ground was initiated in 2002 by a local South African mining company. This has been implemented in an attempt to deal with adverse rock conditions by

[8] South Africa Mine Accident Statisics from Mine Health and Safety Council

looking at mine design and mine layout. Numerical modelling systems were used in considering the size of the support pillar, stope dimension and extraction sequence. These measures are taken to reduce damage to the rock during mining because mining involves fracturing rock structures or working with already fractured rock. In 2003 the mine design and layout phase conditions were controlled. All types of mine support standards were replaced by electronic-based systems, which could be incorporated into computer-aided design plans and made accessible by all mines.

Molina et al. contributed to safety in underground mines by using wireless networks as monitoring systems for the detection of dangerous gases and collapses [38]. Their system is used for early warning signs to prevent injuries and significant economic losses. This contribution is the preparedness phase in the prevention of disaster, as shown in Figure 2.4. On the other hand, Stormont et al. proposed using autonomous robots to manage risk in disaster scenarios. One of the scenarios presented in this thesis is the robots deployed for rescue at the World Trade Centre disaster site in 2001. The authors also discussed the semi-autonomous exploration rover that explored the planet Mars and the computer model used to investigate issues of trust and the impact of reliability in a fire-fighting scenario [39]. In the fire-fighting scenario, the robots are called upon for assistance when the human fire-fighters are not making enough progress in putting out the fires. The model was simulated and created using a Net Logo agent modelling language.

Research focusing on reducing rock fall and rock burst accidents in underground mines has continued since 2006. A local South African mining company is co-sponsoring a project with a South African research institute called the Integrated Damage Rheology Model (IDRM), which is focusing on numerical modeling of mining and seismic data [40].

It is commonly said that prevention is better than cure. Rather than spending millions of US dollars on deploying a solution in a disaster rescue, it is better to spend that money on prevention (safety).

Figure 2.4: The four phases of a disaster

According to the US Federal Emergency Management Agency (FEMA), disaster has four phases, as depicted in Figure 2.4, which the public associates with immediate life-saving or mitigation efforts, leading to disaster response or disaster management[9]. This research is envisaged to contribute to the first phase of a disaster, which is prevention. This will be achieved by sending interacting autonomous multi-robots to inspect the safety of a-mine by assessing the status of the rocks, roofs, gases and water level before miners enter to perform operations. Mine inspections before miners enter the underground area to work have not really been explored. Such inspections are part of what we are proposing as preventive safety measures.

Safety is a major element in the underground mine. Despite a significant reduction through safety research measures, the number of disasters in underground mines in South Africa and the world at large remains high [15]. The development of a robot that can identify hazardous materials such as toxic canisters and mitigate incidents of chemical release has been described in [94]. For decades, the use of robots to inspect tasks in harsh and dangerous environments has been discussed.

Mulder et al.[70] described the use of mobile sensor networks for inspection tasks in a harsh industrial environment. PicoSmart, the mobile sensing project, has been the context in which the research is carried out. The rationale of PicoSmart [70] is to deploy a swarm of autonomous robots as a mobile sensor network. The robots maintain and repair a faulted pipeline from the inside since they have additional computational and maneuvering capabilities. An integral approach has been taken to deal with the distributed coordination problem for mobile sensor networks in industrial inspection tasks. Simulations were used to enable the mobile sensors to learn at different levels of the network.

[9] Computing for Disaster: http://cra.org/ccc/docs/init/computingfordisasters.pdf (accessed on 07 May 2014).

The drift-type environment has been investigated using an integrated system for autonomous exploration and navigation. The system functionality includes vehicle localization and wall-following [79] steering through an intersection controller. The results prove that the system is reliable.

Investigation of underground mine communication was a major focus in [15]. Major issues that underground mine communication systems must take into account were also investigated and discussed and communication types, methods and their significance were presented. Underground mining involves mine-to-ground communication and in-mine communication. It was observed that since each communication type comes with its own problems, it is extremely difficult to come up with a single system that can provide solutions to all of them simultaneously. The authors believe that the similarities between underground mines and disaster scenarious will lead to more interesting communication applications in future.

2.1.3 Robotics-Related Disasters

Robotics accidents result from equipment malfunction, poor operating practices and other elements, as depicted in Figure 2.5[34]. Over the years industrial robot safety has received extensive attention, followed by robots outside factories, such as domestic and hospital robots. Recently, hazardous area robots have been employed in cases where safety and reliability are critical. The types of breakdown that affect robots' reliability and safety, according to [34], are human error, software failure, random component failure and systematic hardware failure.

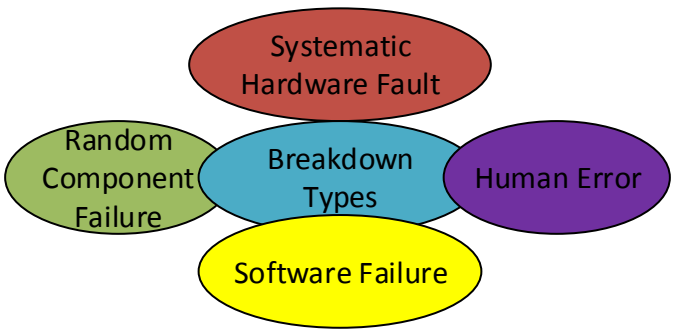


Figure 2.5: Types of breakdown that affect robots' reliability and safety

Software failures are the most important concern in robot-related accidents. This is because they determine the flexibility and robustness of the robot. Human error is also part of total robot breakdown, which is caused by people who design, manufacture, test, operate and maintain robots. Unpredicted breakdowns are regarded as random component failures [34]. Unknown mechanisms in robot designs are major causes of systematic hardware faults.

Over the years, industrial robots have caused some hazards. The following are some past robot-related accident data from [35]:

- An injury led to the death of a worker who violated rules for safety devices by entering a material-handling robot area. The worker was trapped between the robot and a post anchored to the floor.
- The first robot-induced fatal accident occurred in Japan in 1978.
- The first fatal robot-related accident in the USA occurred in 1984.
- Between 1978 and 1984, at least five fatal accidents involving robots occurred; four of these accidents occurred in Japan and one in the USA.
- Line workers and personnel were at the greatest risk of injury, according to a study of 32 robot-related accidents in Japan and the USA in 1987.

To understand the rationale for this work better, we conducted a safety survey by sampling articles on behavioural models from IEEE and science direct databases. Fig. 2.6 shows the findings from the empirical survey. About 20 articles from 1990 to the present date were reviewed for this survey. The outcome shows that while research efforts have been undertaken to mitigate some safety-related issues, more work still needs to be done to deal with safety issues, especially in underground mines. The larger percentage of the existing work needs improvements; about five out of the 20 surveyed articles do not need improvement and 5% of the surveyed articles did not refer to safety issues in both robotics and mining. For instance, in the mining sector, several rules and regulations have been issued to miners to ensure safety in the mine, but the more the rules, the more the disasters

and fatalities. Dhilion [36] states that mine managements and regulators should stop creating more rules.

Human beings cannot keep up with all these rules for safety; they are prone to errors most of the time, especially in confined environments such as mines [32]. This has created urgency to build models for MRSs to mitigate the dangers in these hazardous environments.

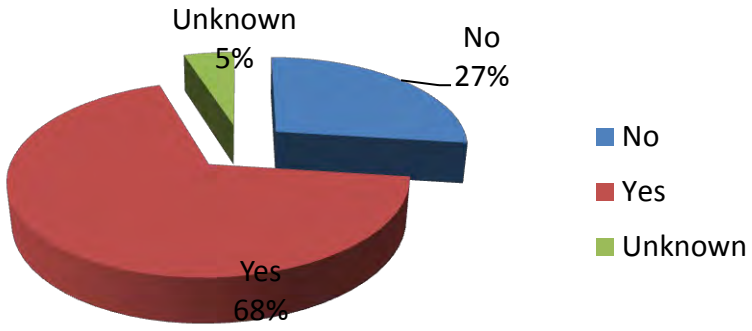


Figure 2.6: Do existing behavioural models need improvement for robotics and/or mine safety?

2.2 Cooperative Behavioural Modelling Framework

Multi-robot systems share the need to cooperate, creating the problem of modeling behaviour. When dealing with multiple robots, with randomness involved, the dynamic and unpredictable nature of the environment has to be considered. Hence, the behavioural modelling system has to cope with the random (dynamic and unpredictable) nature of the system. Researchers, on the other hand, have been captivated by this cooperative and coordination problem of MRS in recent times. A list of literature on multiple robots' cooperation implemented in space was reviewed in [42]. Using multiple robots to achieve tasks has been more effective compared to using a single robot. See for instance [17, 59] (and all references therein) for some specific robotic tasks. Kudelski et al. designed a framework for realistic simulation of networked MRS [90]. In their work, networked robotics was the core area. This area integrates MRS and network technology in order to achieve communication behaviour among the robots.

A multi-robot planning algorithm for tunnel and corridor environments is implemented in [60]. The overall problem formulation is implemented using a topological graph and spanning representation.

Activities such as a single robot drive, differential robots drive that can rotate in place, etc. are carried out at different positions on the graph. The algorithm as presented assumed a centralized planning architecture. Researchers further compared the multi-robot planning with a sequential planner. Their future work is to consider decentralized planner architecture and they might explore hybridizing the two planning algorithms. Different methods have been used to tackle coordination of MRS in different domains [62]. Complete task coordination by multi-robots was handled [61, 59]. An extension of a market-based approach was used. This was achieved by generalizing task descriptions into tasks trees, thereby allowing tasks to be traded in a market setting at a variable level of abstraction.

Thorpe et al. discussed a starter on field robots [63]. According to their discussion, field robotics involves the automation of platforms such as air, sea and land in harsh unstructured environments such as underwater exploration, mining, agriculture, highways, etc. Field robots are made up of three parts: navigation and sensing, planning and control, and safety. Their work also discussed the challenges and progress of field robots. One of the major challenges of field robots in harsh environments such as an underground mine is the problem of position determination (localization). This is so because the global positioning system can only help in an environment where sky views are guaranteed. However, progress has been made in automating some of the environments that are cooperatively constrained.

Parker et al. [6] identified a need in MRS to develop a mechanism that would enable robot teams to generate cooperative behaviours autonomously. The cooperative multi-robot observation of multiple moving target application was presented as a rich domain for studying the issues of multi-robot learning of new behaviours. The need for learning and adaptation was identified and revealed from their implementation results.

An investigation into automating the underground mine environment after blasting, called “making safe”, was carried out in [64] to ensure the safety of the environment after blasting. Blasting in an

underground mine is the controlled use of explosives to excavate, break down or remove rock. The need to investigate the stability of the environment after blasting before any mining operation takes place is of the highest priority, hence, the reason for automation. The automation was centred on a persistent area of concern in South African underground mine operation called hanging walls, which is caused by rock burst and fall of ground. There are also other persistent areas such as the levels of TG that pose great disaster threats to the lives of miners, for instance heat sicknesses, explosions, pneumoconiosis[10], etc. Some of these disasters might result in fatalities and/or disabilities. Again, when an accident happens during mining operations, rescuers find it difficult to respond to accidents immediately. Looking at the aforementioned concerns, there is a need to create models for safety inspection of underground mine operations. For instance, monitoring the underground mine environment for detecting hazardous gases and/or smoke should be one of the important safety measures. Continuous monitoring of workers and equipment is another crucial safety measure [15]. Picking up the sound from roof cracking to monitor when a roof is about to fall is also a safety item.

The proposed cooperative behavioural model presented in [65] promised to handle the safety part of field robots presented by [63] in underground mines. Their model architecture has three layers; the first layer handles the cooperative behaviour of the model, the second layer deals with the scalability degree of the model, while the last layer handles the applicability of the model. This thesis is built on [65]. The need to extend cooperative behaviour of MRS to underground terrains has been mentioned. Dangerous safety inspection of the mine by humans immediately after blasting can be replaced by inspection by multi-robots. This we believe will reduce the dangers faced by miners and mine inspectors.

While mine safety rules and training have been used to guide underground miners in the past and in recent times, the use of autonomous robots is gradually being introduced [64]. However, a robust model that can guide MRS to achieve a safe environment for miners before mining operations

[10] Occupational safety and health - fall of ground management in South Africa, SAMRASS - code book for mines

resume is yet to emerge. Thus, it is an ongoing research challenge. Our model is built by hybridizing two artificial intelligence algorithms called QL and ACS.

The cooperative behaviour of MRS has been studied by some researchers, but its practicality and implementation have not been finalized. Much research is still required in this area to achieve cooperation in real life. Generally, two types of cooperation exist [42]:

- **Passive Cooperation:** In this case, cooperation exists when the whole environment is observed, which is called emergent behaviour. Robots that sense one another as obstacles and navigate away from one another are examples of this type.

- **Active Cooperation:** Robots in this group cooperate through any of the means of communication (wired, radio, wireless, etc.). In this type of cooperation, details of the action to be performed by the robots needs to be communicated to all the robots.

One can use unmanned ground vehicles as an example of the distinction between active and passive cooperation. Active cooperation exists between vehicles with sufficient bandwidth on the communication channel for negotiation of actions, while passive cooperation exist in situations where there is limited bandwidth and so no negotiation with one another occurs. MRS has been classified using different definitions by different authors. Dudek et al. present the taxonomy for MRSs in [17].

- **Task-based classification:** This explains the type of task a group of robots can undertake. For instance, some (i) require multi-agents, (ii) are traditional multi-agents (transportation, industrial, agricultural and fishing-related tasks), (iii) are inherently single agents, and (iv) may benefit from the use of multiple agents.

- **Size of robot collectives:** This explains the number of robots that can be in a group. Dudek summarised this as: (i) the minimal collective is made up of one robot (SIZE-AIONE), (ii) the simplest group is made up

of two robots (SIZE-PAIR), (iii) there could be a limited number of multiple robots (SIZE-LIM) and (iv) an infinite number of robots (SIZE-INF).

- **Communication:** This is a class that shows the different ways in which robots interact with other robots and the environment. In this class, there are three subclasses: (i) a situation where there are no direct communications between robots (COM-NONE), (ii) robots communicating locally (COM-NEAR) and (iii) communication of robots in a wider range (COM-INF).

- **Reconfiguration:** This shows the rate at which the group can re-organise itself and move with respect to one another. The arrangement can be (i) static (ARR-STATIC), which means that the topology is fixed, (ii) a coordinated rearrangement (ARR-COMM); here the topology can be alternated, and (iii) dynamic (ARR-DYN); behaviours can change with changing environment.

- **Composition:** This explains the different units from which the group is made up. The collective can be made up of different units: (i) identical (CMP-IDENT), (ii) homogeneous (CMP-HOM) and (iii) heterogeneous (CMP-HET).

- **Control:** The existing architectures are explained here. They are (i) centralised (CTL-CEN), (ii) decentralised (CTL-DEC) and (iii) hybrid (CTL-HYB) [42].

A novel economic approach for coordinating robots based on the free-market system was introduced in [96]. The free-market approach defines revenue and cost functions across the possible plans for executing a specified task. The steps are: determining revenue and costs, function “trev” and “tcost” , the role of price and the bidding process, cooperative versus competition, self-organization and learning and adaptation. The architecture eliminates the distinction between benevolent and self-interested agents, since robots can increase their personal profits, eliminating global waste and inefficiencies. The result showed that the robots’ profits increased in proportion to their contribution to the optimization process.

Real-life cooperative hunting by MRS in an unknown environment was investigated in [71]. A novel approach based on a bio-inspired neural network was proposed to handle the evaders and deal with the unknown and changing environment. The task was achieved with limited sensory information and little communication burden. The approach used in this paper can be applied to other cooperative tasks, such as fire disaster response and robot policemen.

Geunho et al. presented a decentralized formation controls for a team of anonymous mobile robots performing a task through cooperation [74]. This was a scenario where robot teams are required to generate and maintain various geometric patterns adapting to environmental changes in many cooperative robotics applications. In particular, all robots must continue to strive to achieve the team's mission, even if some members fail to perform their role. They also presented the formation control architecture and algorithm needed to coordinate multiple robot movements within a team. This was achieved by developing a software framework for supporting general purpose applications of cooperative robots through running the same algorithm.

A framework and algorithms for solving real-time task and path planning problems by combining evolutionary computation-based techniques with a market-based planning architecture were proposed in [97]. In their work, algorithms and techniques to dynamically allocate tasks and to find a set of paths for a team of heterogeneous autonomous vehicles were developed. Task planning, which includes task allocation and scheduling, and path planning were the two sub-problems handled by the algorithms.

2.3 Deploying Cooperative Behavioural Models in MRS Navigation

The use of behaviours as the underlying control representation provides useful encoding that lends robustness to control and allows abstraction for handling scaling in learning as of key importance to

multi-robot systems [95]. One of the methods used in this paper is a survey of a collection of BBS-based approaches to single and especially multi-robot learning that have tackled real-world challenges by taking advantage of the behavioural substrate both at the representational and algorithmic levels. However, much work remains to be done both in the theoretical analysis and empirical use of behaviours and BBS.

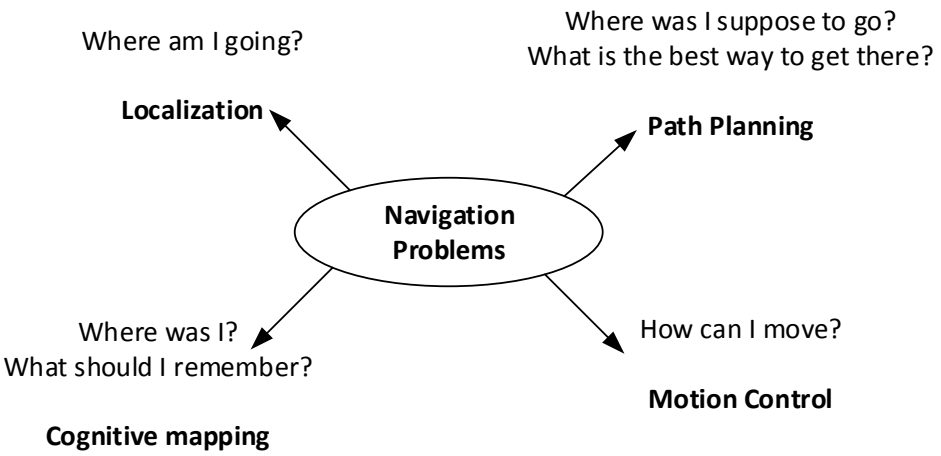


Figure 2.7: Robot navigation problems

Figure 2.7 [84] displayed the robot navigation problems categorized into four, namely 1) localization, 2) path planning, 3) motion control and 4) cognitive mapping. Path planning is one of the navigation processes that have received most attention. In dealing with path planning, the global path can be planned offline before the robot starts to move, as far as the knowledge of the environment is known. When robots avoid obstacles in a real-time environment, this is referred to local path [99]. In [84], ant colony optimization (ACO) was compared to genetic algorithm (GA) and their findings show that ACO can find the optimal path faster within a smaller number of generations than GA. ACO has also been combined with various methods in [85, 86, 87] to achieve different new methods for solving optimization problems.

Our proposed software architecture for a cooperative behavioural model is a fully distributed architecture that uses adaptive action selection to achieve cooperation. Robots in this system are

designed using behaviour-based approaches, where a number of task-achieving behaviours are active simultaneously. The robots in this system react quickly to the actions of other robots, by performing a task differently from the other robots' task and turning in a direction different from the direction of the other robots. To detect and interpret the actions of other robots, a communication network exists in the form of wireless communication between the robots. The robots broadcast statements of their current actions to one another. Based on communication strategy, the use of cooperative behavioural models can be categorized in this thesis as synchronous and asynchronous in MRS navigations.

2.3.1 Synchronous Approach in MRS Navigation

The synchronous approach in MRS navigation involves the use of a learning capability attached to individual robots for navigation to their destinations at the same time. It also involves real-time communication and collaboration between robots as they simultaneously navigate to achieve a given task.

2.3.2 Asynchronous Approach in MRS Navigation

In this case, robots start navigation at different times but connect according to their own schedule. The interaction between the robots is easily captured, shared and distributed in this approach. The ultimate goal for a team of robots is to complete their mission as quickly as possible without wasting energy. This goal is worth investigating using both the synchronous and asynchronous modes of navigation.

The new behaviour-based model proposed in this work involves the use of behaviours as the basic representation level for learning. Qlearning algorithm was first used to test the behaviours of two robots. When compared to cooperative and uncooperative versions, the evaluation indicates that the quality of the solution does not improve in terms of number of iterations, time and the calculated routes. The solution improved effectively when QL was hybridized with ACO. The solution gave better communication, and better output of time, iteration and memory usage when two robots inspection is compared with single robot inspection.

2.4 Different Cooperative Behavioural Models

This section is divided into three subsections. Different approaches/methods used in building cooperative behavioural models, different domains and platforms are explained in 2.4.1 to 2.4.3. There are different architectures for MRS models. Collaborative navigation algorithms [76, 78] are some of the existing architectures. These architectures enable robots to cope with the ubiquitous presence of various types of uncertainties in their environment [77, 72]. Coordinated control models for MRS have also attracted some attention in recent times [91, 93].

2.4.1 Behavioural Models Based on Collision Avoidance Success

Collision avoidance involves methods that attempt to avoid simultaneous access to the same resource, such as scheduling of timeslots, randomised access times and carrier detection schemes. Different methods have been used by researchers to achieve collision avoidance in behavioural-based methods. A step-forward approach built on an omni-directional vision system was implemented to avoid static obstacles and dynamic obstacles by [43]. Three different algorithms were implemented in this work: (i) the algorithm for avoiding static obstacles, (ii) the algorithm for avoiding dynamic obstacles and (iii) the algorithm for avoiding static and dynamic objects. This approach has helped in decision-making on collision avoidance in MRS. A sparse edge detection and reconstruction algorithm has confirmed a more detailed view of the environment, which aids in collision avoidance with obstacles for unmanned aerial vehicles [44]. A test flight was carried out to test the algorithm in a more realistic scenario. The test confirmed that the sparse edge reconstruction algorithm resulted in a much more detailed view of the environment than if a single, more detailed edge detector had been used. The results also show that the sparse edge reconstruction algorithm has a higher signal-to-noise ratio than if stereoscopic correlation is carried out once after applying a single-edge detector.

2.4.2 Behavioural Models Based on Intelligence

Intelligence is a fundament tool for any successful behavioural model. Cooperative model based intelligence has been achieved in different domains and environments. Some of the intelligence applied to cooperative behavioural models involved using a finite state machine (FSM) principle on cooperation capture for MRS [4]. The authors state that FSM was used because it simplifies the design process of the system and also achieves maximum results with little effort. Several solutions for how robots cope with uncertainties along the path from interpreting raw sensor inputs to behaviour selection were found using autonomous colour calibration, illumination invariance and autonomous sensor and actuator modelling [9]. A group of intelligent robots work cooperatively to transport an object to a goal location and orientation in an unknown dynamic environment [73]. In this work, obstacles may be present and even appear randomly during the transportation process. Robot control, multi-agent approaches and machine learning were integrated into the developed physical platform to cope with the main challenges of the problem. The empirical results show that the algorithms are particularly suitable for decision-making in MRS with resource or behaviour conflicts.

2.4.3 Behavioural Models Based on Scalability

Scalability[11] is the ability of a system to handle a growing amount of work. Yasuda et al. [23] listed one of the benefits of MRS as the ability to add and remove a robot from the system. A study of scalability properties revealed that the ability to increase the number of robots is needed in hazardous environments and in certain domains where there are high probabilities that robots may suffer damage [20]. It was further stated that a group of robots is likely to finish one task quickly and robustly. One can see that the proposed model is sophisticated, in view of its socio-economic impact, scalability, and cooperative behaviour.

[11] en.wikipedia.org/wiki/**Scalability**

Four distributed algorithms for assigning swarms of homogeneous robots to subgroups to meet a specified global task distribution were presented in [98, 75]. The four algorithms are robust to the removal or addition of robots at any time. There are some problems with the algorithms due to bandwidth limitations in the communication hardware of the swarmBot system.

2.5 Open Research Issues in Behavioural Modelling for Autonomous MRS

Some of the open research issues in literature after critical survey and investigations are depicted in Figure 2.8. The open research issues are scalability, a hazardous environment and real-life application.

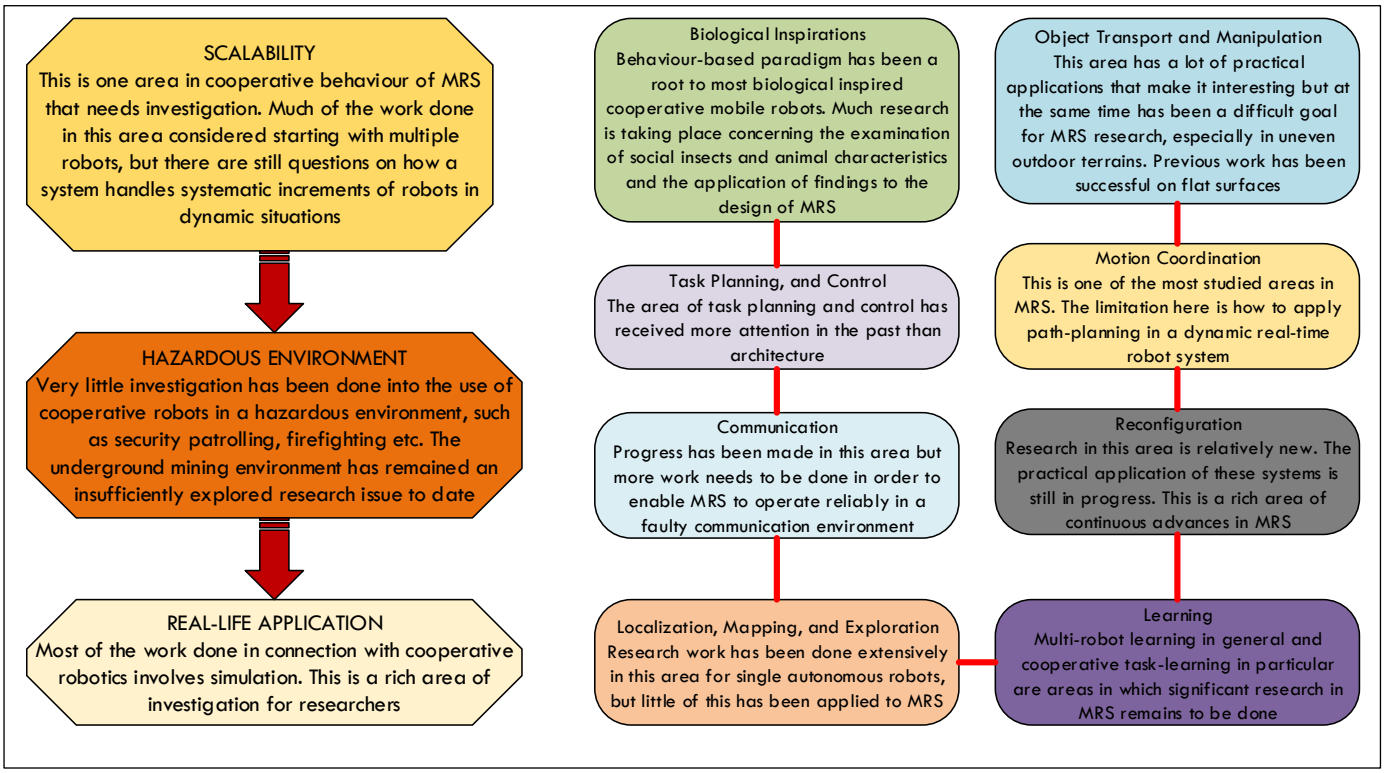


Figure 2.8a: MRS open research issues on cooperative behaviour

Figure 2.8b: MRS research topics and their key open issues

2.6 Machine Learning

Robots require intelligence to solve problems. Learning is central to intelligence. It is necessary for robots to acquire knowledge, since intelligence requires knowledge. According to [95], learning is a desirable virtue for a multi-robot system, because it is what helps the robot to cope with a dynamic or unknown environment, find the optimal cooperation strategy and make the entire system

increasingly flexible and autonomous. Although the majority of the existing commercial multi-robot systems are controlled remotely by a human, the autonomous performance will be the objective of next generation systems. Hence, without a learning capability, it will be almost impossible for a robotic system to become a fully autonomous system. This has made the introduction of machine-learning technologies a necessity in multi-robot domains. Machine learning (ML) serves this purpose. Machine learning is a system that is capable of acquiring and integrating knowledge. The machine-learning system can continuously self-improve owing to the capability to learn from experience, training, analytical observations etc. and thus exhibits efficiency and effectiveness. Figure 2.9 explains the start and end of a machine-learning system.

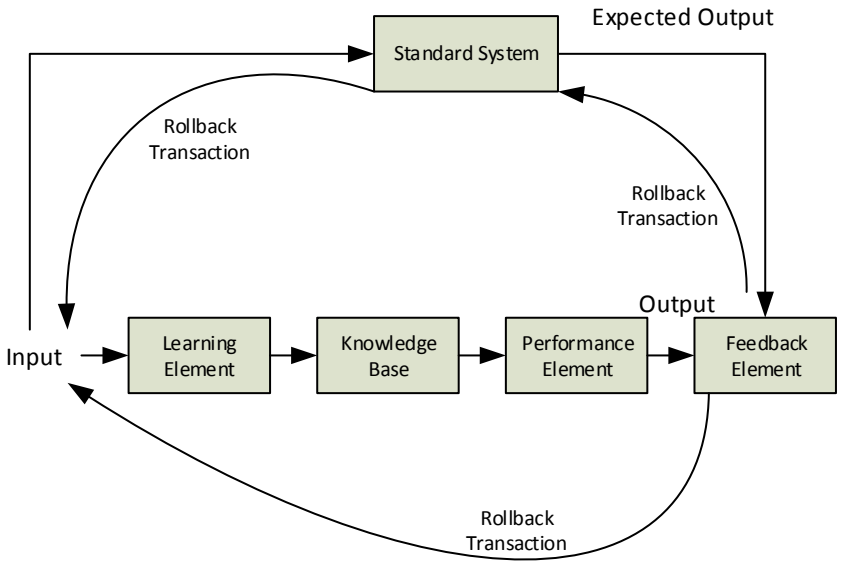


Figure 2.9: Learning system model

A ML system starts with some knowledge and a corresponding knowledge organization to be able to interpret, analyse and test the knowledge acquired. The elements indicated in Figure 2.9 are the components of a learning system model. The learning element receives and processes the input from the environment. The knowledge base is like the database where information received is added or replaces the existing knowledge. With the information received, the system is updated and the corresponding output is produced. The feedback element receives the standard from the two points, and the learning element is used to determine what should be done in order to produce the correct output. The idealized system or the standard system is used as a check for the new system. Transaction rollbacks handle the loops when the feedback element is wrong. The success of a

machine-learning system depends on the algorithms. These algorithms control the search to find and build the knowledge structures. Some research works done with ML techniques were reviewed and we conducted a survey to show whether the ML techniques are appropriate for space and collision avoidance problems. Figure 2.10 summarizes the results obtained from the survey. Out of 25 publications surveyed from IEEE and science direct databases, 17 suggested that ML systems are safe enough to avoid collision in any environment; five of the publications disagreed, while three were indifferent about using a machine-learning system for collision avoidance. The work on a robot soccer player platform done in [46] and [47] revealed that

ML techniques are safe to avoid collision. [46] applied coupled agents on modular QL to enable the robot soccer system to avoid collisions. Furthermore, [47] used the support vector machine and multilayer perceptron techniques to develop a teamserver program that supports collision avoidance in MRS.

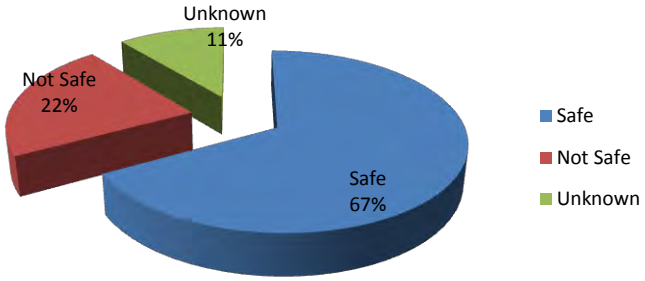


Figure 2.10: Are machine learning systems safe enough to avoid collisions?

2.6.1 Partially Observable Markov Decision Processes

The Partially Observable Markov Decision Process (POMDP) is used in modelling many kinds of real-world problems such as robot navigation problems, planning under uncertainty etc. It is used where access to a state is not available but information about it is obtained through the observation model. It is a generalisation of the Markov Decision Process (MDP). MDPs are modelled mathematical frameworks used in making decisions in cases where outcomes are partly random and partly under the control of a decision-maker. We use POMDP for modelling because it is a standard framework for stochastic processes and it also captures process and sensor uncertainty. POMDP is a special case

of MDP[12]. For MDP, the environment is fully observable while it is partially observable for POMDP. The agent in POMDP cannot execute the optimal policy, since the current state is not necessarily known. Equation 2.1 defines POMDP. Here, agents can represent the situation of the environment through the belief state. A belief state (b) is a probability distribution over s. $b(s)$ is the probability of being in state s when the belief state is b. Subsequently b can be calculated from a previous b. The following defines a POMDP:

- Set of states s, set of actions A , set of observations O.
- Transition model $T(s, a, s')$.
- Reward model $R(s)$.
- Observation model $O(s, o)$ – probability of observing observation *s* in state o.
- Optimal action that depends on the agent's current belief state.

POMDP can be discrete or continuous. Given a belief state, an agent can perform an action a and perceive observation o , and the new belief state becomes

$$-b's' = \alpha O(s', o)\Sigma T(s, a, s')b(s) \qquad (2.1)$$

Many real-world problems are not fully observable and the Markov assumption is often effective. The proposed problem of this research is not an exception.

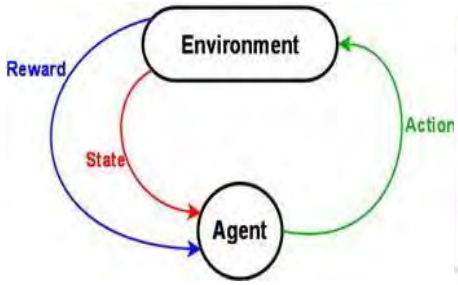
2.6.2 Reinforcement Learning

To develop cooperative behaviours for these robots, collaboration is required. Cooperation and collaboration are two fundamental elements required in behaviour-based robotics [42]. These elements are adapted from biological mechanisms. For instance, the study of cooperative behaviour of bees and ants from swarm intelligence (SI) shows the possibilities for simple robots to work together and solve a very complex problem [48]. Here, the model is designed to help the robots to

[12] POMDPs for Dummies: http://www.cs.brown.edu/research/ai/pomdp/tutorial/index.html (accessed on 09 May 2014).

learn, make decisions, execute actions and in the end get a reward. The model that is developed will be used to discover the best behaviour to handle safety inspections in underground mines by MRS.

Reinforcement learning (RL) is used to learn how to behave in a new environment. Here, autonomous agents learn to act optimally/best without human intervention. Agents learn by randomly interacting with their environment and getting sporadic rewards. The purpose is to maximize these rewards. The reinforcement algorithms selectively retain the outputs that maximize the received reward over time. The learning system must prefer the best experienced actions in order to accumulate many rewards. However, to discover better action selections for the future, it has to try new actions. Figure 2.11 explains the repeated sequence of events and the cause and effect of an RL cycle. An RL system has four sub-elements apart from the agent and the environment: policy, a reward function, a value function and/or a model of the environment.



1.	The agent observes an input state
2.	An action is determined by a decision-making function (policy)
3.	The action is performed
4.	The agent receives a scalar reward or reinforcement from the environment
5.	Information about the reward given for that state/action pair is recorded

Figure 2.11: Reinforcement learning cycle with cause and effect

RL problems are made up of four elements: a set of states, a set of actions for each state, a transition function that specifies the probability of transitioning from one state to another when performing each action and a reward function, which indicates the amount of reward or cost associated with each transition [67].

Learning is needed in applications where humans are unable to explain their skills, for example in speech recognition, where solutions change over time, for instance in routing on computer networks, and where a solution needs to be adapted to a particular case, for example user biometrics . There are three main types of machine learning: supervised learning, unsupervised learning and reinforcement learning. The platform that is used in developing our model consists of two autonomous

robots that know each other's actions through collaboration. For instance, before the first R1 could take an action, it needs to know the action taken by the second R2 and vice versa. Learning is required to address this cooperative problem and this can be undertaken by reinforcement learning algorithms and swarm intelligence techniques.

Models imitate behaviours of the environment, predict the next state and give the initial state and action. Models are generally used in planning future situations. Figure 2.12 is an example of learning in robotics, where the small round lemon objects represent static navigational features, the shaded rectangular objects represent obstacles, the wheeled objects represent multi-robots and the green spherical object represents the goal state.

Figure 2.12: Reinforcement learning in robotics

In the scenario, if the agent hits an obstacle, it gets a negative reward, if it gets to the goal, it gets a positive reward and when it gets to the goal faster, it gets a bigger reward. A list of some of the RL algorithms is shown in Figure 2.13 with two different categories: the model-based and the model-free. Model-based ones build a model of the environment and use it to learn the environment. Examples include adaptive dynamic programming and Monte Carlo methods. The model-free ones learn the policy without a model; the advantage of this method is that it requires limited memory for its operation. The temporal difference method is a type of model-free RL.

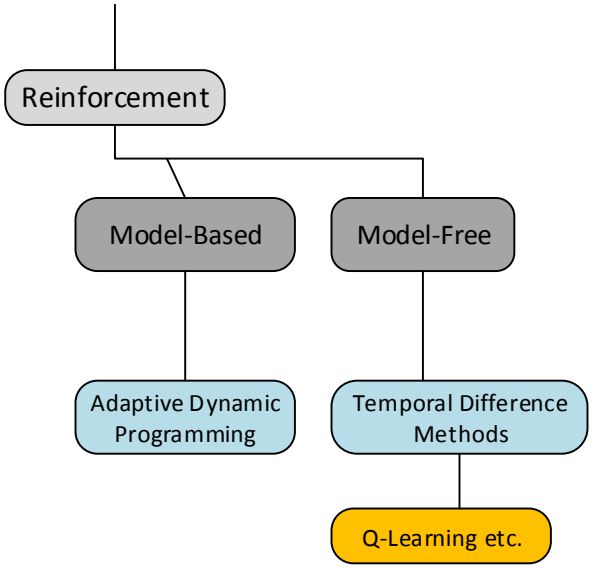


Figure 2.13: List of reinforcement learning

In the last decade, a class of approach in which the agent learns based on reward and punishment it receives from the environment, called reinforcement learning, has become the methodology of choice for learning in a variety of domains, especially the robotic domain [66, 92]. An RL is one of the artificial intelligence algorithms that can achieve learning by experience. This enhances robots' interaction with the environment. The focus of the work [66] was motivated by the challenges of learning in noisy, dynamic environments, such as the ones in which mobile robots exist. Its particular interest is in the complex case of concurrent multi-robot learning. Minimizing the learning space through the use of behaviours and conditions was one of its interests, as well as dealing with the credit assignment problem through shaped reinforcement in the form of heterogeneous reinforcement functions and progress estimators. Experimental results showed that in the given test domain, shaped reinforcement in the form of heterogeneous reward functions and progress estimators based on multi-modal sensory feedback were crucial, given the complexity of the environment-robot and robot-robot interactions.

We investigate the use of RL to assist the behaviour of an MRS in safety inspection of underground mines. At any time step each robot is in a specific state in relation to the environment and can take one of the following actions: inspect, ignore or shut down. Each robot receives feedback after

performing an action, which explains the impact of the action in relation to achieving the goal. The effect of the action can be either a good or bad reward. This reward is measured in terms of values. Therefore, the value of taking an action a in any state s of the underground terrain is measured using the *Action-Value function* called Qvalue $Q^{\pi}(s,a)$. When a robot is starting from state s, taking action a, and using a policy pi (π), an expected return, which is defined as the sum of the discounted rewards, is achieved.

2.6.3 Q-Learning Algorithm

In this research, QL, a method of RL, is explored. The purpose of RL methods is to study $Q^{\pi}(s,a)$ values so as to achieve optimal actions in the states. QL is an on-line RL method that requires no model for its application and stores the reinforcement values outcome in a look-up table. The QL architecture used in this work consists of learning threads, which amount to the number of robots involved in the inspection behavioural task. Each robot in the learning thread carries out QL in the environment. Table 2.3 explains the QL algorithm used in the behavioural model.

Table 2.3: Q-Learning algorithm

Steps:	**English Expression of the Steps**
initialize $Q(s,a)$ *arbitrarily* *repeat (for each episode):* *initialize* s *Repeat (for each step of episode):* *Choose* a *from* s *using policy derived from* Q *Take action* a *, observe* s *,* s' $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ $s \leftarrow s'$ *until s is terminal*	1. Initialize the Q-values table, **Q(s, a)**. 2. Observe the current state, **s**. 3. Choose an action, **a**, for that state. 4. Take the action, and observe the reward, **r**, as well as the new state, **s'**. 5. Update the Q-value for the state using the observed reward and the maximum reward possible for the next state. The updating is done according to the formula and parameters described on the left-hand side. 6. Set the state to the new state, and repeat the process until a terminal state is reached.

α is the learning rate set between 0 and 1. At 0, Q-values are never updated, hence nothing is learned; learning can occur quickly at 1. γ is the discount rate set between 0 and 1 as well. This models the fact that the future rewards are worth less than the immediate rewards. $max_{a'}$ is the

maximum reward that is attainable in the state following the current state; that means the reward for taking the optimal action thereafter.

QL is a competitive and search-based algorithm inspired by computational theory. It is not necessarily a multi-agent algorithm but can be adapted to a multi-agent or multi-goal scenario. The success of this algorithm relies on the value and policy iterations, which can be adjusted by some unfairness (heuristics) to fit the current problem scenario. The most competitive action is selected by its value and action leads to another state or condition. Both value and policy are updated after each decision. Harnessing QL for an MRS scenario increases the cost exponentially and the overall performance drops in the same direction. As the robots increase cost, such as completion time, memory usage and awareness factor (other robots in the environment), search time increases. However, following our heuristic model of QL, which was mainly determined by the policy we set to achieve our goal, our QL performs well above traditional QL.

2.7 Swarm Intelligence

SI is a component of computational intelligence that is used in solving stochastic/probabilistic problems, for example in the field of robotics/artificial intelligence (AI), process optimization, telecommunication and entertainment [50]. In the field of robotic/AI, SI can be employed to address the cooperative behaviour of robots because of the emergent collective intelligence of groups of simple agents. Group foraging of social insects and cooperative transportation[13] are some of the examples of emergent collective intelligence. ACO, particle swarm optimisation (PSO) and bee colony optimisation (BCO) are some of the techniques of SI, as depicted in Figure 2.14. SI is interesting in solving the aforementioned problems because of the following features: distributed system of interacting autonomous agents, self-organised control and cooperation, indirect interaction, performance optimisation and robustness. These features are inspired by nature and modelled through the enablement of biological species (bees, ants, particles etc.).

[13] Swarm Intelligence-Introduction http://staff.washington.edu/paymana/swarm/krink_01.pdf (accessed on 09 May 2014).

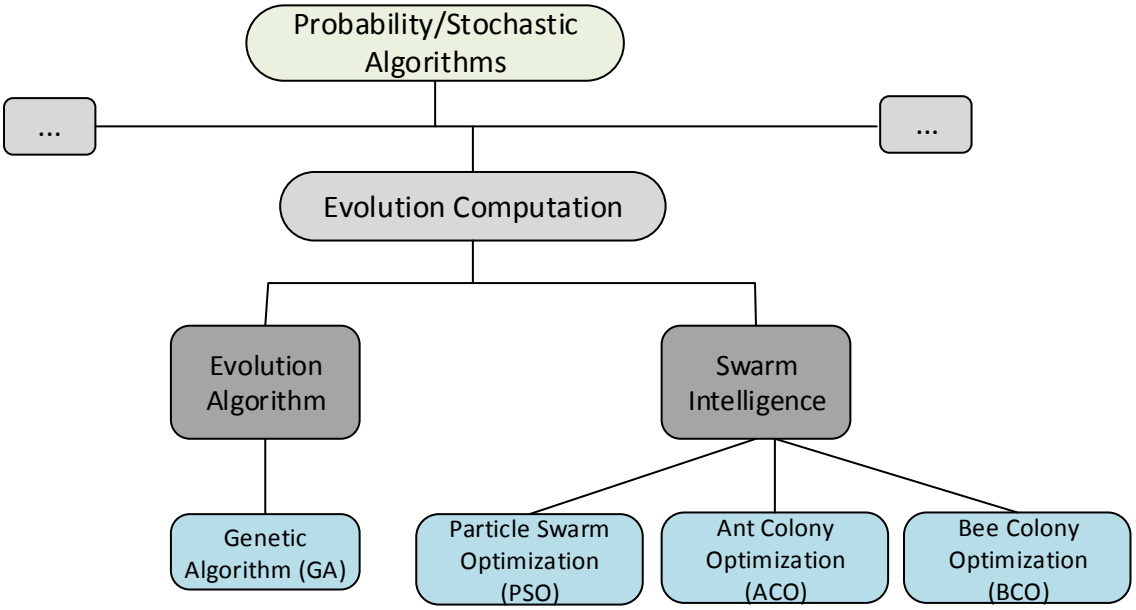


Figure 2.14: The taxonomy of stochastic algorithms

The stochastic nature of swarm robots is one of the deficiencies of swarm behaviours. Ondrej et al. presented a working architecture in [88] that improved the performance of search and rescue operations using fuzzy manual control. [89] reviewed relevant literature and research in the area of animal behaviour. Based on certain observations, several useful emergent behaviours have been developed as a result of a number of proposed primitive reflexive behaviours.

2.7.1 Ant Colony Algorithms

ACO is a type of swarm intelligence. Bonabeau et al. [52] define swarm intelligence as any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies. This implies that anytime something is inspired by swarms, it is called swarm intelligence. Researchers have been thrilled by swarms because of some of their fascinating features. For instance, the coordinated manner in which insect colonies work, notwithstanding having no single member of the swarm in control, the coordinated ways in which termites build giant structures and how the flocks move as one body and the harmonized ways in which ants quickly and efficiently search for food can only be attributed to an emergent phenomenon [53, 54].

Ants, an example of a social insect colony, achieve their self-organizing, robust and flexible nature not by central control but by stigmergy. Stigmergy, also known as a pheromone trail, describes the indirect communication that exists within the ant colony. The indirect communication that is triggered by pheromone trails helps in recruitment of more ants to the system. Ants also use pheromones to find the shortest paths to food sources. Pheromone evaporation prevents stagnation, which also helps to avoid premature convergence on a less than optimal path [55].

ACO is a multi-agent-based algorithm. The first implementation of this optimization algorithm is in a classical search-based (combinatory) problem, the travelling salesman problem, giving the shortest path to a specified destination. After this feat, many researchers have used it or its variants to model different problems. In this work, a variant of ACO is used to find the optimal path for MRS. Table 2.4 describes variants of ACO and their meaning [56]; see also [57] and all references therein.

Table 2.4: Variant of ACO

S/N	Year	Algorithm
1.	1991	Ant System (AS)
2.	1992	Elitist A.S
3.	1995	Ant-Q
4.	1996	Ant Colony System
5.	1996	Max-Min A.S (MMAS)
6.	1997	Ranked Based A.S
7.	1999	ANTS
8.	2000	BWAS
9.	2001	Hyper-Cube A.S

In AS, the pheromones are updated by all the ants that complete a tour. ACS is the modified version of AS, which introduces the pseudorandom proportional rule. In elitist AS, ants that belong to the edges of the global best tour get an additional amount of pheromone during the pheromone update. MMAS introduces an upper and lower bound to the values of the pheromones trails. All the solutions are ranked to conform to the ants' length in ranked-based AS [58]. Shaogang et al. simplified a path-planning problem by minimizing the path (grid method) that connects every point.

They achieved this using the ant colony algorithm [80]. ACO has been applied to routing and load-balancing [81]. Ant-Q algorithms were inspired by work on the ant system. There are some similarities between Ant-Q and QL [82]. Gambardella et al. applied Ant-Q to the solution of symmetric and asymmetric instances of the travelling salesman problem.

2.7.2 Ant Colony System

Our interest is in implementing an ACS to find the best possible round trip inspection path in our model environment. This best possible inspection path will be supplied as input or made available for the robots using QL for inspection decisions. In most of the route-finding scenarios, the nodes are linearly joined and are not largely distributed. This is a scenario where ants can forage along at least two paths (multi-goal path environment, though in our case there is a path exposed to four different goals; see section 4).

ACS is derived from AS. The AS is the first ACO algorithm proposed in the literature by [68]. In AS, an ant k being in the node i chooses the next node h with a probability given by the random proportional rule defined as

$$P(i,h) = \frac{[\tau_{i,h}]^{\alpha}\,[\eta_{i,h}]^{\beta}}{\Sigma_{n\in N^k}[\tau_{i,h}]^{\alpha}\,[\eta_{i,h}]^{\beta}} \qquad (2.2)$$

where N^k is its feasible neighbourhood. $P(i,h)$ is the Probability of moving from *i* to h, α is the Pheromone influence factor, β is the influence of adjacent node distance and $\tau_{i,h}$ is the pheromone concentration of node $n_{i,h}$. Once an ant has visited all nodes, it returns to its starting node. Solutions are in the form of paths through the problem graph. Each ant adds one vertex to its path at each step during construction [49]. In ACO algorithms, the ant will move from vertex I to vertex h with a probability calculated as Equation 2.2.

As Equation 2.2 gives a probability of selecting a given edge, a method of utilizing this information is required to actually determine the next step in an ant's solution. Typically this takes the form of roulette selection [49, 51]. In roulette solution, all options are given a probability of being selected,

and then a random number in the range (0, 1) is drawn. The probabilities of selecting each option are added to a sum one after another. If adding a given probability to the sum increases it beyond the random number then the corresponding option is selected. In ACO, this means that the selected edge is added to the ant's solution.

An improved ant colony optimization is applied in solving the robot path planning problem by [83]. Once each ant in the population has constructed a solution, the second phase known as the pheromone update, takes place. The pheromone update can be separated into two steps: evaporation and deposit. In the evaporation step, the amount of pheromone removed is calculated as

$$\tau_{i,h} = (1-\rho)\tau_{i,h} \tag{2.3}$$

where $\tau_{i,h}$ is the amount of pheromone on edge i,h, ρ is a parameter that controls the rate of evaporation. This formula is applied globally to all edges in the graph. The second step, deposit , is calculated for each ant over the path it took through the graph, and typically takes the form:

$$\Delta\tau_{i,h} = \begin{cases} \frac{1}{c_k}, & if\ ant\ travels\ edge_{i,h} \\ 0, & otherwise \end{cases} \tag{2.4}$$

where c_k is the cost of the kth ant's solution. By iteratively applying Equations 2.2, 2.3, and 2.4, the amount of pheromone will accumulate on edges which appear to be parts of good solutions to the problem thus attracting future ants towards those edges.

2.8 Chapter Summary

This chapter reviewed the related literature in robotics and underground mine safety. Several approaches to addressing MRS, underground mine safety, cooperative behaviours of robots were discussed. The underlying principles behind the intelligence of the proposed topic of this thesis were explained. Robotics and underground mine disasters were discussed and the gap that our proposed model is bridging was clearly shown.

3. Proposed Behavioural-Based Model for MRS

3.1 Introduction

The following are some of the factors considered in the course of building the model. Each robot has to learn to adjust to the unknown underground mine environment. In this case, each robot requires intelligence and a suitable machine-learning algorithm is adopted. No robot has global information on the environment because of its limited sensing capabilities. Each robot has limited communication capabilities; therefore each robot has to keep track of the information of others to remain in a team. Figure 3.1 describes the proposed framework as an approach for building the distributed, coordinated inspecting model for MRS.

3.2 Proposed Cooperative MRS Behavioural Framework

This work develops a cooperative behavioural model that can guide an autonomous MRS into achieving inspection tasks in an underground terrain. Figure 3.1 depicts the proposed hybrid framework of the model. The framework is divided into three layers: the cooperative behavioural layer (bottom layer), the scalability layer (middle layer) and the application layer (topmost layer).
The learning capability of the MRS is achieved in the bottom layer, with the reinforcement learning algorithm and swarm intelligence technique. This intelligence enables R1 to take action knowing the action of R2 and vice versa. At the middle layer, scalability in terms of the number of robots the system can accommodate is achieved. This is expedient because a team of robots tends to achieve tasks more quickly and effectively than single robots. This scalability is handled with some memory management techniques, as indicated in Figure 3.1. The real-life implementation is achieved by using the information acquired from the topmost layer. Figure 3.2 is the breakdown of the framework in Figure 3.1.

The model proposed in this research deals with the way in which robots need to find their way within communication range and uses a broadcast approach for effective communication of navigation status. There is a base-station or server that serves as a backup for the information captured and

analysed from individual robots. A team of robots cooperatively inspecting an area in the underground mine will need to know where they are and where to go next, so it is obviously a continuing problem and our contribution in this case is that before R1 takes action, it broadcasts its location and inspection status to other robots, R2, R3, etc., and vice versa. An unreachable robot receives packets of information based on the destination address through re-routing from the nearer robots. The reliability of this broadcast method is due to the ability to determine the extent of the task executed already by looking at the memory of any leading robot in the team.

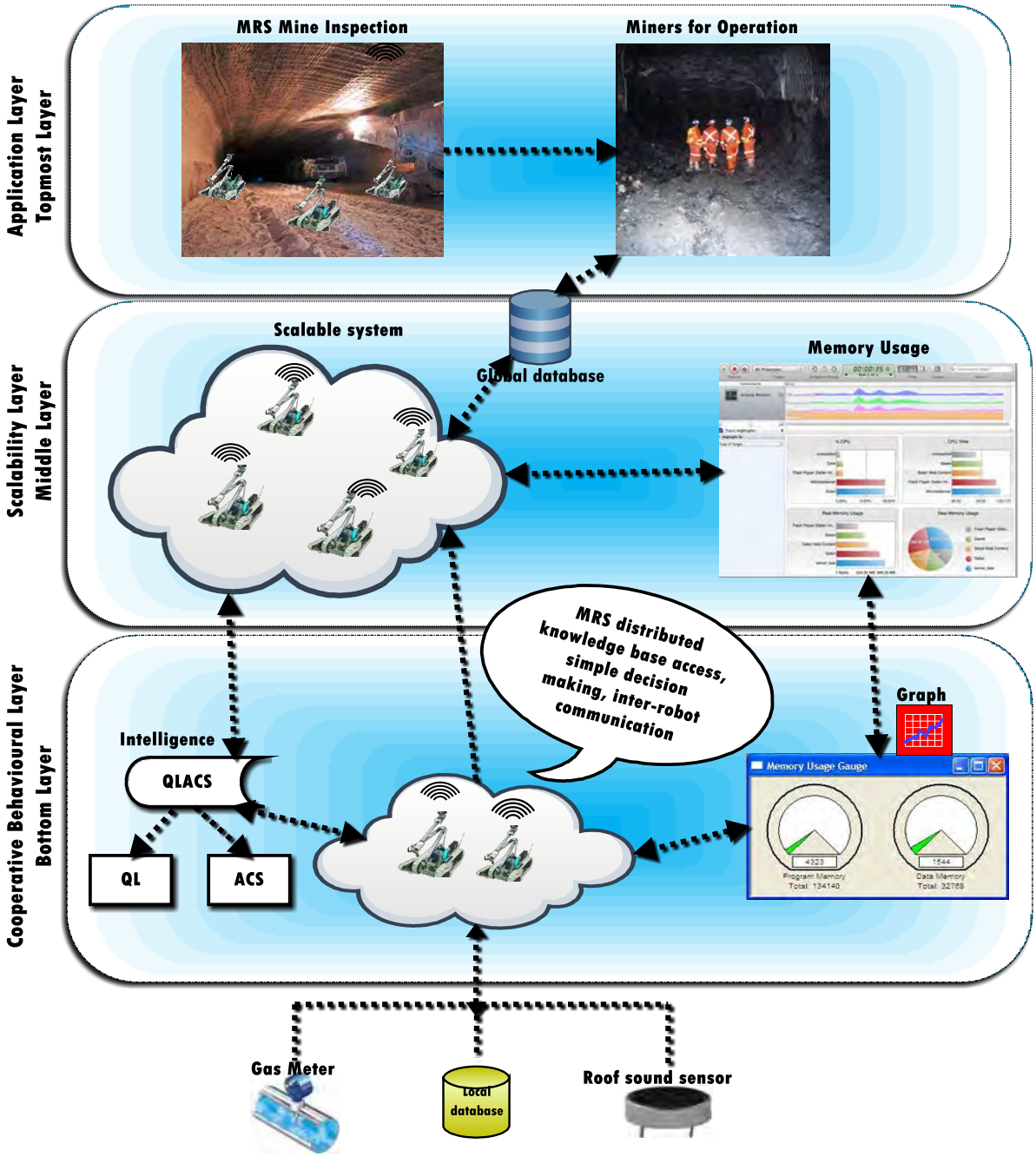


Figure 3.1 Framework of the proposed QLACS model for cooperative behaviours

The cooperative and route-finding behaviours of the multi-robots are handled in the bottom layer. In this layer, two robots learn and adjust to the environment with the help of an intelligent hybrid model designed using the QL and ACS algorithm. Their cooperative behaviours are tested with the model in the layer. Then, with an increment in the size of the robots, the scalability features of the model are verified at the middle layer. Both the bottom and middle layers are tested for memory usage and time consumption in the course of the experiment. The results acquired from the bottom and middle layers are stored for future use in the topmost layer.

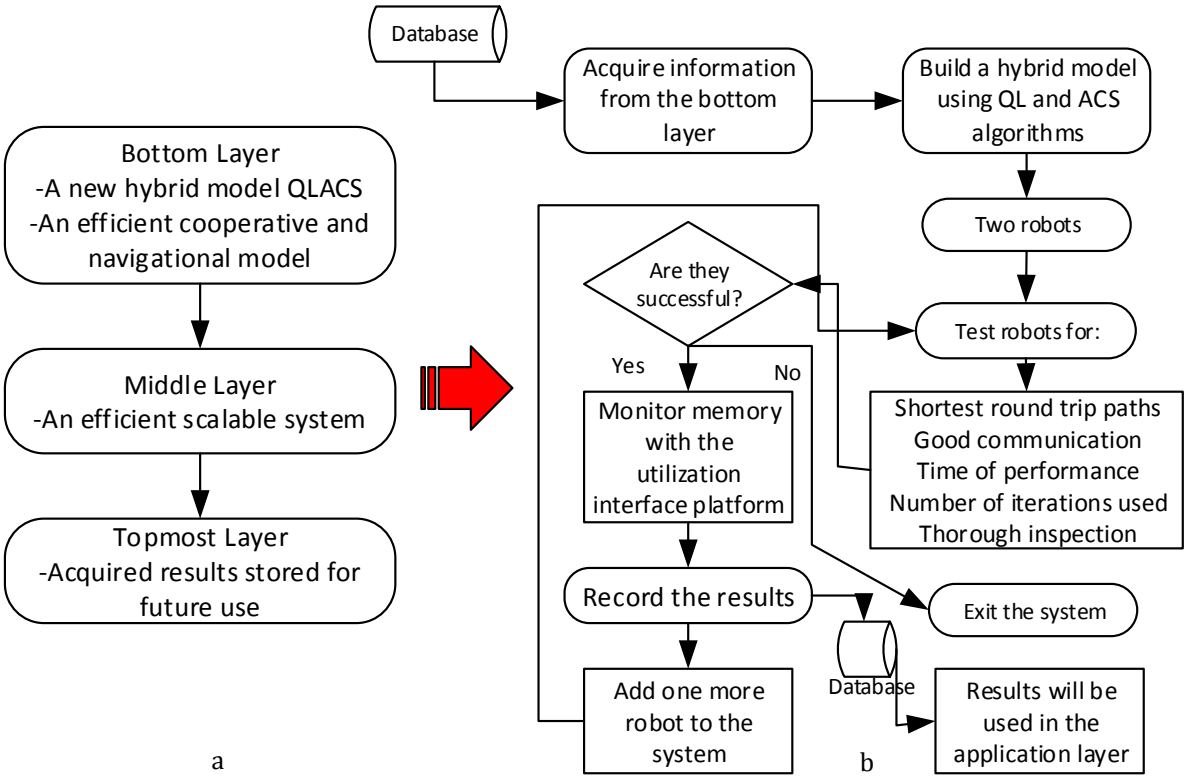


Figure 3.2: Breakdown of the framework. (a) contributions of the framework, layer by layer and (b) processes of the multi-robot behavioural system

There are different definitions of scalability depending on the background of the person defining it, the technology being considered, and the operational use of the technology [69]. In robotics, there are different types of scalability: (i) size (ii) memory (iii) power (iv) information, etc. However, in this research, we intend to focus scalability on the number of robots (size) that can lead to safety inspection of underground mines. This is expedient because teams of robots tend to achieve tasks more quickly and effectively than single robots. Rosenfeld et al. [45] surveyed a good number of studies on how performance scales with an increasing number of robots in a group and their corresponding changes in productivity.

Our work contributes to scalability in underground mines by monitoring how the memory of the system handles the addition of more robots to a team using the memory management scheme called paging (virtual memory). Again, we conducted a survey on 25 publications from IEEE and science direct databases on scalability to determine how these can contribute to safety in underground mines. Figure 3.3 shows that 62% of the work done in scalability will support safety, 20% will not and 18% is undecided. This is supported by [22], which states that using multiple robots to assist in dangerous real-life scenarios such as in mines, security patrolling jobs and rescue operations helps in saving human lives. The authors' states that it is better for humans to monitor tasks done in hazardous environments from a safe location as robots get the work done. However, [23] demonstrates safety in scalability by adding and deleting robots in the system. Damaged robots are deleted from the system to create room for new robots to be added to enhance productivity and safety.

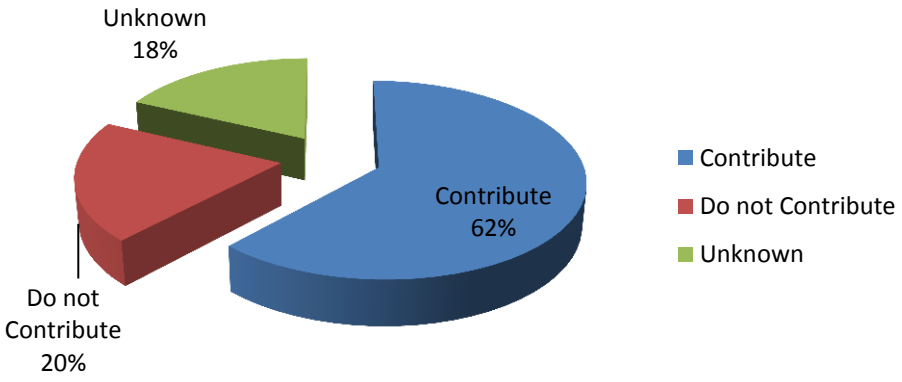


Figure 3.3: Does scalability contribute to safety?

3.3 Problem Formulation

Suppose we have seven rooms/states connected by doors/links representing underground mine regions, as shown in Figure 3.4 and labeled as shown in Table 3.1. We label the rooms A to F. The outside of the mine can be thought of as one big room (H). Notice that doors F and C lead outside the mine H. We put two robots in rooms F and C as their starting states respectively. The robots inspect one state at a time, considering the obstacles encountered in the process. The robots can change direction freely, having links/doors to other rooms/states. In each of the states in Figure 3.4 two actions are possible: inspection of roof cracks (RC) and level of TG, or ignoring the state, as it

has been inspected earlier. According to our proposed model, a robot can inspect at least half of the given underground mine region to attain its maximum performance, which in turn attracts a good reward. When the current state of a robot is the end point of the inspection task, the robots exit the mine using the exit points C and F respectively. The possible transition states of the robots are displayed using the state diagram in Figure 3.5. The state diagram and the transition matrix are formed using the QL algorithm.

The global map is assumed to be known by the robots but there is no prior knowledge of the local map. Robots only know what they have sensed themselves and what their teammates communicate to them. Not only does our improved QL depend on the map of the environment, but each robot learns through experience about local changes. They explore and inspect from state to state until they get to their goal states. Our proposed model QLACS achieves an offline mode for the route-finding algorithm (ACS). This means that the global view of the map would have been provided before the learning robots start.

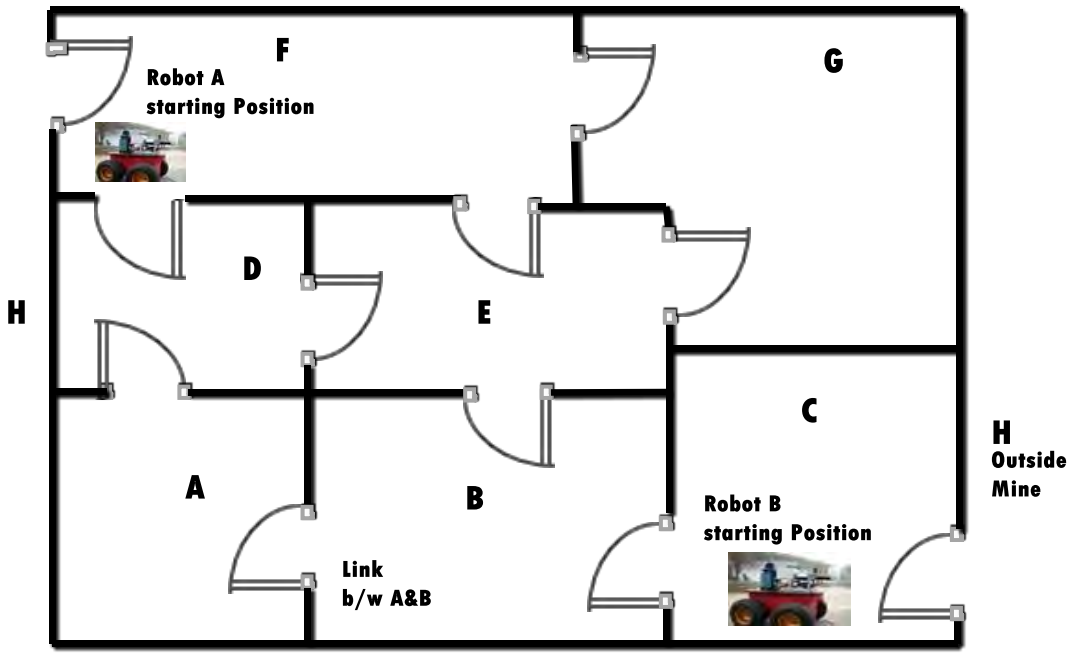


Figure 3.4: Model of the environment with two entrances and exits (2EE)

Table 3.1: State and possible actions of the environment

	State	Possible Actions
1	A (Lower Left Part (LLP))	Inspect, Ignore
2	B (Lower Middle Part (LMP))	Inspect, Ignore
3	C (Lower Right Part (LRP))	Inspect, Ignore
4	D (Middle Left Part (MLP))	Inspect, Ignore
5	E (Central Part (MCP))	Inspect, Ignore
6	F (Upper Left Part (ULP))	Inspect, Ignore
7	G (Upper Right Part (URP))	Inspect, Ignore
8	H (Outside Mine Part(OMP))	Shutdown

Analysis by a state transition diagram is presented in Figure 3.5. The possible state actions of the robots are presented in Table 3.2. The states of the robots are reduced as follows: searching or inspecting for RC and TG levels in each state or room, recording the outcome of the inspection in the robots' memories. The processes involved in achieving good communication while the robots inspect the states are broadcasting inspected states to the other robots and ignoring the inspected/broadcasted state, avoiding collision with obstacles and other robots and finally moving to the next available state that has not been inspected. Figure 3.5 shows states C and F as the two starting points from outside (H). At state F, the red arrows show the possible states the robot can navigate to. The arrow heads show where the robots can navigate to from any state they are. The green arrows show the states that can be navigated to from state E, the same goes for all other states. (50,100) represent the rewards that can be obtained from inspecting or ignoring a state.

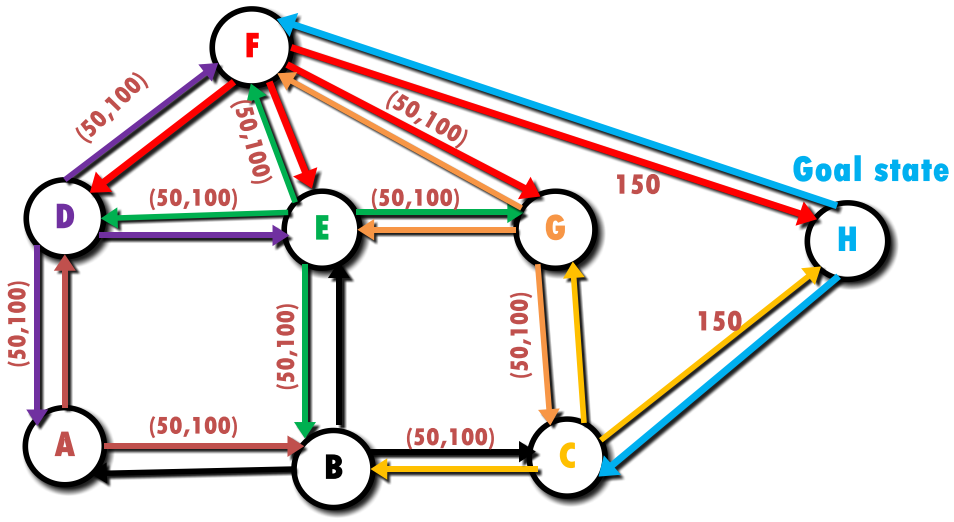


Figure 3.5: Events and transition diagram of the modelled environment with H as goal state

Table 3.2: Initial reward matrix

Robot's State	Robot's Action							
	A	B	C	D	E	F	G	H
A	-	50,100	-	50, 100	-	-	-	-
B	50,100	-	50, 100	-	50,100	-	-	-
C	-	50,100	-	-	-	-	50,100	150
D	50,100	-	-	-	50,100	50,100	-	-
E	-	50,100	-	50,100	-	50,100	50,100	-
F	-	-	-	50,100	50,100	-	50,100	150
G	-	-	50,100	-	50,100	50,100	-	-
H	-	-	50,100	-	-	50,100	-	-

The QL algorithm is used to determine what action is to be selected in any particular state. Let the action (a) = {inspect, ignore, shutdown}, state space (s)= {dimensions in the topological map}, sensor readings(s_r), hazardous conditions (H_c) = {RC, TG}. Each robot is configured with an aerial scanner and chemical-sensitive sensor that will provide readings(s_r), to determine if there is a hazardous condition, H_c, in the environment. The selection of an action by a robot through the QL algorithm is based on the information from the broadcast. The whole framework uses a decentralized QL scheme such that each robot has its own thread with its QL and table. All Q-values on the table of the robots in the team are initialized to zero, i.e. states corresponding to such positions have not been inspected. When a robot enters a new state it broadcasts its information for the current state to all other robots in the environment. The broadcast indicates whether the current state has been inspected or not. The broadcast is a search of corresponding positions in the memories of all visible robots. If the resultant search returns a zero; the broadcast indicates that the state has not been inspected and if it returns a value greater than zero it indicates that the state is not available for inspection. All robots must receive a broadcast before they act in any particular state. When a robot gets to a state, it receives a broadcast and passes its value to the QL algorithm to make a decision. The robot carries out the decision of the QL algorithm. The policy of the QL algorithm makes a robot carry out an inspection if the resultant search of the broadcast returns zero, and ignores it if the resultant search is greater than zero. A robot only shuts down if all states have been visited and inspected. As the broadcast information is passed to the QL algorithm, the policy is iterated towards an optimal value and condition.

The broadcast avoids the cost of multiple inspections and the QL ensures that robots take appropriate actions. The only state in which inspection is carried out would have sensor readings (s_r), which indicate H_c. For taking an action (a) in state (s), the robot gets a reward (R), which is used in QL to compute the Q-value for the current state, and can send a broadcast to other robots. Figure 3.5 and table 3.2 show the restrictions and possible transitions among node points, indicating the possible rewards for any particular action (a) in any state (s). Every other transition besides the goal state could result in a reward of 50 or 100 and at completion the reward is the maximum, 150. We consider it wasteful to make the robots scan first before sharing intelligence because such action would slow them down and make them expend more energy. Robots are to provide the inspection results, showing actions taken in different states and the nature of conditions detected in any particular state. The introduction of the broadcast approach to determine the team's exploration reduces the execution time and energy cost of the whole team and makes the collaboration effective. So in a multi-robot scenario the task can be effectively executed if the robots are made to share intelligence as they progress. Robots do not have to waste time and energy in doing the same task already carried out by other robots in the team.

GIVEN: On a mathematical explanation of the above, suppose a safety pre-inspection of toxic gases or rock fall or some combination of the two is being carried out on a piece of complex underground terrain in Figure 1.5, say $L\,Km^2$, there is a limited number of MRS with different capacities, R, and precious inspection time, T minutes. Every region/state x_1 in the terrain requires a capacity of robot R_1 of MRS and limited time T_1 while every state x_n requires robot R_n of MRS and inspection time, T_n. Let P_1 be the positive reward of QL for correct behaviour on state x_1 and P_n be the reward on state x_n. This research aims to maximise positive rewards by choosing optimal values for states $x_1 \ldots x_n$ as follows:

Maximise: $P_1.x_1 + P_2.x_2 + \cdots + P_n.x_n$ (objective function)

Subject to constraints: $x_1 + x_2 + \cdots + x_n \leq L$ (limited total states)

$R_1.x_1 + R_2.x_2 + \cdots + R_n.x_n \leq R$ (limited MRS capacity)

$T_1.x_1 + T_2.x_2 + \cdots + T_n.x_n \leq T$ (limited inspection time)

Non-negativity constraints: $x_1 \geq 0,\ x_2 \geq 0, \ldots, x_n \geq 0$ (MRS cannot inspect negative states)

In terms of solving this optimization problem, we use the proposed QLACS model to compare the time and number of states inspected. The number of robots used is also compared. The graphs results in sections 4 and 5 also give more insight into the solution of the problem.

3.4 Basic Navigation and Cooperative Behavioural Models

QLACS has two components. The first component is formed by an improved ACS and the second component is formed by an improved QL. The improvement occurs because some heuristics were added to the ordinary QL and ACS to achieve the hybrid QLACS. However, the second component of QLACS, which is an improved QL, was initially used to solve the proposed problem. After much analysis, we realized that the system needed to be optimized for effective cooperation and communication.

Using the second component of QLACS to solve the basic navigation and cooperative behaviour, the possible actions were set for each robot as inspect, ignore and shutdown (after reaching the goal state H). Also, a reward system that would reflect the possible actions of the robots was chosen. In other words, a robot gets 150 points only when the goal is achieved (shutdown), 100 points for ignoring an already inspected area (ignore) and 50 points for inspecting an uninspected area (inspect). Figure 3.5 shows the transition events of the model and Table 3.2 displays the possible state action for each robot. The way the QLACS second component works here is based on both navigation and communication behaviours.

Achieving navigational behaviour with the second component of QLACS has some cost associated with it. In our scenario, because we want the robots to share intelligence by broadcasting results (robots search through other robots' memory), our problem is not solely navigational, but also cooperative. Our behavioural actions are inspect, ignore and shutdown. We noted that these actions of interest are not navigation oriented; there is no way we could use them in making decisions on transition. The functions for decision can be integrated to assist the others. Therefore, our behavioural model is an integration of two behaviours: (1) navigational behaviour and (2) cooperating behaviour through decision-making. The integration works with a route-finding method called *RandomStateSelector,* which we introduced in the second component of QLACS to help determine where the robot goes from the starting point to the exit point. Two parts of the *RandomStateSelector* method are introduced in this work. The first one is the *RandomStateSelector_C_H,* which is used to transit from state C to H and the second one, *RandomStateSelector_F_H,* transits from state F to H. This method works, but not effectively, because some of the states are repeated several times because of the random selection method. However, the decision part of this second component of QLACS, which is handled by a method called *CheckInspection,* works efficiently. *CheckInspection* is responsible for sharing the broadcast among the agents. The broadcast looks at all the stored Q-values on all the robots and returns a signal for the action that needs to be taken. Therefore, we concluded that the heuristically accelerated component of QLACS has proven to be effective by showing evidence of effective communication in making inspection decisions using our model. It did not guarantee the shortest possible time for inspection because of repeated state decisions. In this light, we only harnessed the communication strength of the second component of QLACS for communication and cooperation. Figure 3.5 and Table 3.2 form the basis of the QLACS second component.

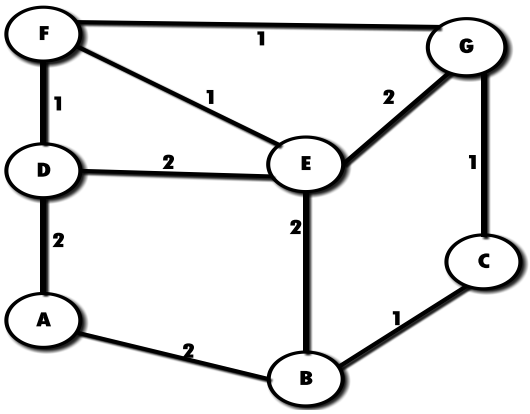


Figure 3.6: Weighted map/graph of the model environment

Table 3.3: Combined adjacency and weight matrix

	F	G	E	D	A	B	C	H
F	0	1	1	1	0	0	0	1
G	1	0	2	0	0	0	1	0
E	1	2	0	2	0	2	0	0
D	1	0	2	0	2	0	0	0
A	0	0	0	2	0	2	0	0
B	0	0	2	0	2	0	1	0
C	0	1	0	0	0	1	0	1
H	1	0	0	0	0	0	1	0

To take care of the random state selector problem encountered in implementing the algorithm used for the second part of QLACS, we introduced an optimized route-finder algorithm. This route-finder algorithm, which forms the first component of QLACS, is a swarm intelligence technique. Figure 3.6 and Table 3.3 form the basis of the exploration space of the agents (ants) that achieved the optimum route finding. The weighted graph in Figure 3.6 is based on the number of obstacles the ants will encounter from the points of entry F and C. The combined table in Table 3.3 contains the weights of the obstacles and evidence of an edge between any two vertices (states). It shows that there is a connection between any two vertices in a graph. Generally, a "1" or "2" depicts the existence of an edge while a "0" represents the absence of an edge, i.e., no transition between such vertices. The constructed graph is an undirected graph, providing evidence of some agents coming from F or C of the mine (logical space). It is unidirectional because agents can move in any particular direction. This means that the same weights on the edges apply to both directions. The graph does not show H; we assume that once the agents reach F or C, they exit if all inspections have been done.

3.5 The Navigation Model

This section explains the navigational formation of the proposed model. Firstly, we constructed a graph with associated weights and created pheromone tables for all states with their probabilities. Figure 3.6 and Table 3.3 form the basis of this construction. The methods used in constructing the class framework for the navigation formation is tabulated in Table 3.4.

Table 3.4: Methods and functions for the ACO framework

Methods	**Functions**
ACO_Framework	This is a class constructor where all class variables are initialized
ConstructGraphWeight	This is a method where the graph is constructed based on the adjacency/weight matrix
StartIndex	This module designates all robots to start from either F or C; so that we can have optimized paths verified from both directions
Distance	Returns the weight between two adjacent states or nodes
AntsInitializationAndUpdate	Ants' trails are initialized and updated in this module
PheromonesInitilization	Pheromones that are secreted in the foraging environment are initialized here to a value slightly greater than zero: 0.01
EdgeInTrail	This method checks if there is an edge between two states or nodes
UpdatePheromones	Pheromone information is updated here as the ants forage
Length	This method sums up the length of all the edges by an ant until the current edge
GetTheNextCity	This method uses the probability values and roulette wheel selection scheme to determine the index of the next state.
GetAcoNextState	This method returns the name for the next state based on the order of the starting points.
TransitionProbability	This method computes the probabilities based on the pheromones, attractiveness and other heuristics to determine the chance for any state among the potential states
GetBestTrail	This method gets the best inspection path from the BestTrail method below.
BestTrail	This method scans through the ants' trails and selects the best (shortest and complete round trip) trails for ants coming from F and C. These best paths will be made available to the robots for inspection. It is noteworthy that these are the best possible paths for round-trip inspection; no other path is considered optimized.
BuildTrail	The method coordinates the foraging activities. Threads are created for each trail and new states are generated until the goal is achieved.
ProcessDisplay	The text formatting and output are done here. A global string variable is loaded with the format (output)
ReturnDisplay	This method returns the string variable from the *ProcessDisplay* method above to Programe.cs class.
Programe.cs	All threads start from here. It is the super controller of all processes. Output file is written to file here.

3.5.1 ACS Model Development

In ACS, construction of a solution uses the pseudo-random proportional rule. This altered the way the probability information from Equation 2.2 (page 43) was used, not how it was generated. A

undirected graph is employed to establish a free state modelling then the ACS algorithm is utilized to optimize the route. Figure 3.7 explain the pseudocode used in achieving an optimized route-finding system in our model [75]. The route-finding system is the first component of our hybrid model that handles how the agents achieve behaviours from one state to another using the ACS algorithm.

```
1. Boolean ForagingStatus = True //Boolean variable indicates completion for al l the threads
2. LogicalSpace, FinishedThreadBuffer as Buffer
3. Read and Initialize edge distances // i.e. graph construction
1. Initialize pheromones
2. Initialize Ants' trails
3. Initialize edge associated probabilities
4. Initialize Trails' starting and terminating indexes i.e. from F or C
5. While (ForagingStatus <> False)//Checks for when all agents have finished & flag is false
Begin
    Create N number of Threads in Parallel
    Assign Current indexes(edges) to all threads in Parallel
IF ( LogicalSpace==FULL & ThreadID) // all edges have been visited by ThreadID
Begin
    Assign the ThreadID to FinishedThreadBuffer // Thread Shuts down
End
Else
Begin
IF ( Edges== exist)     // connection between two edges i.e. states; values read from file
Begin
    Compute States transition probabilities
    [Compute cumulative of probabilities and use Roulette Wheel (Fitness Proportionate) selection to determine Nextstate]
    Set Nextstate as the Current State
    Build Ants' Trails based on their Unique thread ID by Updating Ants' Trails
    Compute Length and use it in Pheromones calculation
    Compute and Update Ants Pheromones
End
End
IF (Count[FinishedThreadBuffer]==NumberofThreads)//Thorough inspection completed
Begin
    ForagingStatus= False     // While Loop invariant fails and learning stops.
    Select the Thread with the Best Trail starting from F
    Select the Thread with the Best Trail starting from C
End
End of While Loop.
```

Figure 3.7: Pseudocode of the route-finding using ACS algorithm

3.6 The Communication Model

The communication part of the proposed model is explained in this section. The QL algorithms handles this part of the model and some of the methods used in achieving the code framework are described in Table 3.5.

Table 3.5: Methods and functions for the cooperative behaviour framework

Methods	**Functions**
Qlearning	This is the class constructor for the QL class. Most of the global variables are initialized here.
GetReward	This method is responsible for returning the reward associated with any particular action selected.
SetLearningRateGamma	This computes the learning rate for any particular agent based on its experience acquired in other states. It is state-dependent.
GetAndSetFrequencyOfState	This retrieves the experience of the robot based on the state visited.
InitializeQvalues	Q-values of all agents are initialized to zero
InitializeRobot_StartingStates	This method designates starting states to all robots, since the model uses (2EE) for effectiveness.
QMaxValue	It searches and returns the best Q-value.
CheckInspection	This method is responsible for sharing of intelligence among the robots: the broadcast. It looks at all corresponding positions across the Q-values stored on all the robots and returns a signal if action had been taken there before. This helps to determine action taken by any agent. All agents must call for this handle.
ComputeQvalue	It computes the update for the Q value.
RandomStateSelector_C_H	It is the random algorithm used for transition coming from state C.
RandomStateSelector_F_H	It is the random algorithm used for transition coming from state F.
LearningRobots	It is the coordinating module (method). It is the brainbox for the policy iteration. Threads are created and assigned starting states and manage their Q-value updates. It ensures that no thread terminates when thorough inspection has not been conducted, while each thread learns individually but shares intelligence.
Selector	It is the module responsible for getting the next state. Each thread calls this module and gets its new state. It is tied to the route-finding algorithm.
PrintOutput	It formats the results and writes them to an output file.

3.6.1 QL Model Development

The QL pseudocode developed in Figure 3.8 consists of learning threads that amount to the number of robots involved in the behavioural inspection tasks. Each robot in the learning thread carries out QL in the environment. The Q-values are checked for and compared with zero to determine when each robot takes an action of inspect, ignore or shutdown. Figure 4.8 explains in detail how the QL algorithm we adopted for our cooperative part of the work was designed. This enhanced better communication with the robots.

```
1. Boolean CompletedFlag = False //Boolean variable indicates
   completion for all the threads
2. Declare CompletedThreadBuffer // Data structure stores Info about
   completed thread
3. Initialize all Qvalues to Zero      //All Qvalues positions are initialized to
   zero
4. While (CompletedFlag <> True)//Checks for when all robots have
   finished & flag is true
   Begin
        Create N number of Threads in Parallel
        Threads get Current States in Parallel
        Threads get Next States and Indexes in Parallel
        Threads compare Qvalues of all corresponding positions of Current
   States (Each Robot Broadcast Qvalue info)

        IF ((Q ==0) & (ThreadNextState <> GoalState))//Checks if a
   particulate state is available
         Begin
          State is Available, Robot with the CurrentThreadID Inspects
          Compute and Update Qvalue
    End
    IF (Q >0) // checks if a state is not available, because an already
   inspected state has Q>0
     Begin
       State is already inspected, Robot with the CurrentThreadID Ignore
       Compute and Update Qvalue
    End
   IF ((Q ==0) & (ThreadNextState == GoalState)) // Checks for goal
   state and shuts down.
     Begin
       Compute and Update Qvalue
       Goal state is reached and Inspection Completed
       Thread with the CurrentThreadID Shutdown
       Store CurrentThreadID in CompletedThreadBuffer
        IF (Count [CompletedThreadBuffer] == NumberOfRobot) //Learning
   stops when this happens
     CompletedFlag= True
   End
End of While Loop.
```

Figure 3.8: Pseudocode of the behavioural model using QL algorithm

3.7 The Hybrid Model Development

The cooperative behavioural model (hybrid model) is an integration of two algorithms: (1) route-finding algorithm and (2) communication and cooperative algorithm. The integration works in the following way: the optimal route finder (ACS) determines where the robot goes from the starting point to the destination, while QL determines what action it takes when it gets to any of the states. During navigation, when a robot meets an obstacle or when the environment changes, it alters the ACS by building a local map (adaptation/dynamism) and broadcast to other robots before it continuous with ACS guide for subsequent movements. This collaboration works effectively because the optimal route finder has been proven to give the best possible transitions and the heuristically accelerated QL has proven to be effective by showing evidence of effective communication in making inspection decisions. Consequently, it guarantees the shortest possible time for inspection in the absence of wasteful inspection decisions. The analytical development of the hybrid model is described below.

The parameters for building the analytical hybrid QLACS (Equations 3.1 to 3.10) model are presented in Tables 3.6 and 3.7.

Table 3.6: A list of parameters for the ACS model

ACO Parameters	**Meaning**
α	Pheromone influence factor
β	Influence of adjacent node distance
ρ	Pheromone evaporation coefficient
Q	Attractiveness constant
e	Visited edge
e'	Edge not visited
L_k	Length tour of ant *k*
τ	Pheromone concentration (amount)
$\boldsymbol{\eta}$	Specific visibility function (attractiveness)
$\Delta\tau^k$	Pheromone concentration by *Kth* ant
$P_r(i,j)$	Probability of moving from *i* to *j*
$D_{i,j}$	Visibility or distance between I and j
f_i	Fitness of individual in a population
P_i	Probability of being selected among f_i
N	Number of individuals in the population
i,j	Denotes any two adjacent nodes in the graph
M_k	Set of unvisited nodes

Table 3.7: A list of parameters for the QL model

QL Parameters	**Meaning**
Q	Q-value update
s	State
a	Action
R	Reward
γ	Learning rate

ACS starts

Computation of edge attractiveness

$$\eta_{i,j} = \frac{1}{D_{i,j}} \quad (3.1)$$

Computation of instantaneous pheromone by ant k

$$\Delta\tau^k = \frac{Q}{L_k} \quad (3.2)$$

Update of pheromone

$$\tau_{i,j} = (1 - \rho) * \tau_{i,j} + \Delta\tau^k{}_{i,j} \quad (3.3)$$

Computation of edge probability

$$P_r(i,j) = \frac{[\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta}{\Sigma_{e'=(i,j)}[\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta} \quad (3.4)$$

Adoption of roulette wheel

$$\text{Cumulative}(P_r(i,j)) = \sum_{i=1}^{N+1} P_r(i,j) \quad (3.5)$$

$$f_i = \frac{\sum_{j=1}^{N} f_j}{N} \quad (3.6)$$

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \quad (3.7)$$

where α is the pheromone influence factor, β is the influence of adjacent node distance, ρ is the pheromone evaporation coefficient, Q is the attractiveness constant, e is the visited edge, e' is the edge not visited , L_k is the length of the tour of ant *k*, τ is the pheromone concentration (amount), $\boldsymbol{\eta}$ is the specific visibility function (attractiveness), $\Delta\tau^k$ is the pheromone concentration by the *Kth* ant, $P_r(i,j)$ is the probability of moving from *i* to j, $D_{i,j}$ is visibility or distance between I and j, f_i is the fitness of individual in a population, P_i is the probability of being selected among f_i, N is the number of an individuals in the population, i,j denotes any two adjacent nodes in the graph, M_k is the set of unvisited nodes. Equations 3.1 to 3.7 build the complete route-finding model. Equations 3.1 to 3.3 are prerequisite to Equation 3.4. Equation 3.4 is prerequisite to roulette wheel selection. At

the end of equation 3.7, new states are selected and the trail is updated. The best path from both directions is selected and used as input in QL.

Roulette wheel determines a selection by computing the cumulative values concerned and generating a random value, which falls into one of these value positions, i.e. positions corresponding to any particular value under investigation. This type of selection does not just pick the highest value from the list, but also uses a stochastic process to arrive at a fair and globally optimal selection case, unlike many selection schemes that may be largely tilted or skewed in one direction and the outcome will be biased by considering only the large values in selection [49]. We may view these other schemes as greedy schemes, because they follow in the short term and consider a locally optimal option, which may not be globally optimal. We compute the probabilities using Equations 3.1- 3.5. Thence we compute the cumulative probability values, before we use a random generation between [0,1]. The random value should be equal to or close to one of the cumulative values; the position with that value is the favoured selection. It has been affirmed over time that roulette wheel selection is better than other similar greedy schemes. It is the random value that is equal or near to the cumulative value that determines the selection. It allows other values that are not high to appear in the selection and it makes the whole selection exercise even.

QL starts

Each robot in its QL thread

Computes its learning rate

$$\gamma = \frac{0.5}{[\,1+Frequency\,(s,a)]} \tag{3.8}$$

Q-values are updated

$$Q(s,a) = R(s,a) + \gamma(\max(Q'(s,a)) \tag{3.9}$$

Making a broadcast (Decision = Inspect/Ignore/Shutdown)

$$Q(s,a) = \begin{cases} Q = 0 \;\; Inspect\ if\ s_j \neq goalstate \\ Q > 0 \;\; Ignore\ if\ s_j \neq goalstate \\ Q \geq 0 \;\; Shutdown\ if\ s_j = goalstate \end{cases} \tag{3.10}$$

where Q is the Q-value update, s is the state, a is action, R is reward, γ is the learning rate. Equation 3.10 marks the end of the hybrid model development. Equations 3.8 to 3.10 are state-dependent. The states are kept in a buffer and then accessed at run time. ACS and QL do not work simultaneously. ACS works to completion and QL takes the final output as its input. ACS is not repeatedly called while QL is working. Equation 3.8 is Gamma, the learning rate, which is always between zero and 1. This equation is calculated based on the frequency of action of each robot in inspecting states.

3.7.1 Pseudocode for QLACS

This framework forms the basis of our cooperative behaviour model for MRS (QLACS). The pseudo-code for the implementation of QLACS is outlined in Figure 3.9.

INPUT: Edge distance(obstacles), pheromones, ants' trail, associated probabilities, starting and terminating indexes i.e. from F or C
OUTPUT: Effective cooperation, inspection and navigation

```
Boolean CompletedFlag = False //Boolean variable indicates completion for all the threads
Declare CompletedThreadBuffer // Data structure stores Info about completed thread
Initialize all Qvalues to Zero      //All Qvalues positions are initialized to zero
Initialize Best Paths From ACO algorithm  // starting from F and C
While (CompletedFlag <> True)//Checks for when all robots have finished and flag is true
Begin
   Create N number of Threads in Parallel
   Threads get Current States in Parallel from ACO algorithm
   Threads get Next States and Indexes in Parallel from ACO algorithm
   Threads compare Qvalues of all corresponding positions of Current States (Each Robot Broadcast Qvalue info)
   IF ((Q ==0) & (ThreadNextState <> GoalState))//Checks if a particulate state is available
    Begin
      State is Available, Robot with the CurrentThreadID Inspects
     Compute and Update Qvalue
  End
IF (Q >0) // checks if a state is not available, because an already inspected state has Q>0
  Begin
     State is already inspected, Robot with the CurrentThreadID Ignore
    Compute and Update Qvalue
 End
IF ((Q ==0) & (ThreadNextState == GoalState)) // Checks for goal state and shuts down.
  Begin
   Compute and Update Qvalue
   Goal state is reached and Inspection Completed
   Thread with the CurrentThreadID Shuts down
```

```
    Store CurrentThreadID in CompletedThreadBuffer
   End
  IF (Count [CompletedThreadBuffer] == NumberOfRobot)//Learning stops when this
  happens
   Begin
    CompletedFlag= True
  End
End of While Loop.
```

Figure 3.9: Pseudocode for QLACS

3.7.2 QLACS Hybrid Approach Evolution

Figure 3.10 is the architecture of hybrid QLACS explaining the handshake between the two components.

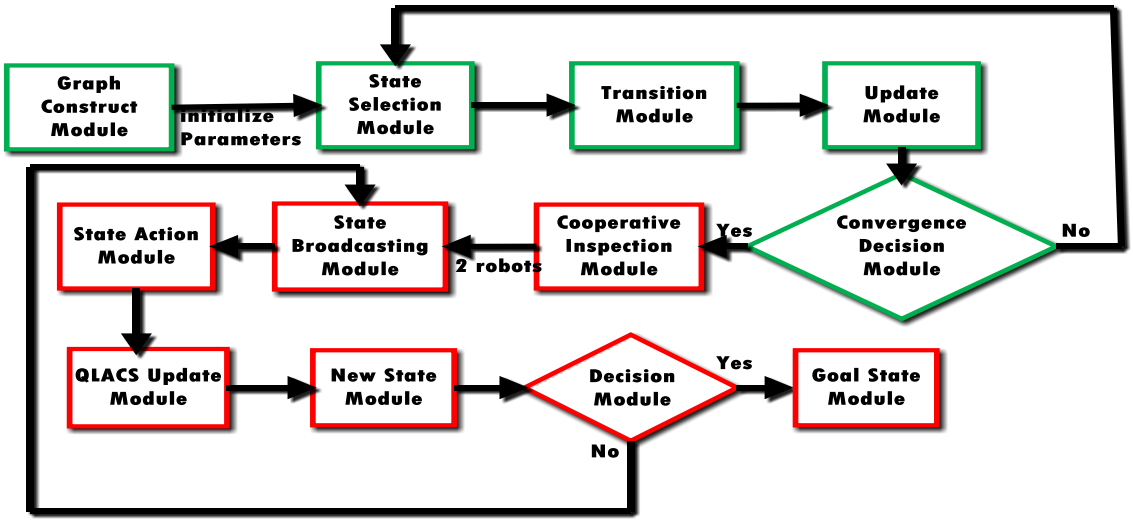


Figure 3.10: Hybrid Architecture

(i) **Graph Construct Module:** This is the interconnection of states in our model environment (see Figure 3.4). It encompasses all possible transitions from F to C and vice versa. Thus from the model we construct an adjacency/weight graph matrix that can be traversed by any graph-oriented algorithm. In our case, there are eight states: primarily seven internal states and a common terminating state. Since the states are not just linear, the environment allows for multiple options, i.e., an agent/ant can choose from any present state. This type of scenario is also called a multi-goal scenario.

(ii) **State Selection Module:** Here, the ants select the start and end states, which according to our model in Figure 3.4 are F and C. These states are selected based on the cumulative probability of two adjacent nodes in Equation 3.5.

(iii) **Transition Module:** This module takes care of the transition rules of the agents by calculating the pheromone concentrations, distances and probabilities using Equation 3.1 through 3.3.

(iv) **Update Module:** After transition from one state to another, an update of the pheromone is computed, after which multi-path planning with the shortest path is achieved.

(v) **Convergence Decision Module:** This is where the best trail/path decision is taken. This is the end product of the first component of QLACS, which is then moved to the second component of QLACS for cooperative behaviour.

(vi) **Cooperative Inspection Module:** This is where the robots are deployed to start inspection. The robots are to use the acquired best paths starting from F and C respectively as input for the second component of QLACS. The two robots are to use the learning rate from Equation 3.8 to learn the environment and use Equation 3.9 for cooperation.

(vii) **State Broadcasting Module:** This module handles the broadcasting behaviours of the two robots; this is achieved by using Equation 3.9. Each robot checks its memory represented by Q-values before taking any decision.

(viii) **State Action Module:** State broadcasting by each robot is immediately followed by action selection. In other words, the state to inspect or ignore is achieved here using Equation 3.10.

(ix) **QLACS Update Module:** After each action selection, the Q-values of each robot are updated using Equation 3.9.

(x) **New State Module:** This module takes care of the robot's new state after updating the Q-values. Each robot runs in its own threads, managing its memory, yet sharing information.

(xi) **Final Decision Module:** This decision module determines if the robot should exit the environment or still do more inspections. It is also controlled by Equation 3.10.

(xii) **Goal State Module:** The completion of the second component of QLACS is getting to the goal state after successful inspection of states. This goal state according to our model in Figure 3.4 is called H.

3.7.3 Communication and Search Cost

The challenge for the cost of communication was anticipated and there are steps taken not to make any assumptions. Initially, the approach for the model adopted QL algorithm which was done by learning with trial runs to perfect and improve the efficiency. What was achieved was Big Oh of upper bound O(n^3)[27] which in the long run after a number of trials got better and better i.e. O(n^3) < O(e^x) . To further improve the O function to a point that is highly accepted, we adopted an algorithm based on the shortest path algorithm. This algorithm is well known for its optimization strength. ACS, a variant of ACO was used to improve the optimization in the model drastically. The complexity achieved from using ACS is O*(nlogn)* [30]. This improved the model to the following:

$ACS = O(nlogn)$, $QL = n^3$, $QLACS = On^3 + O(nlogn)$, $QLACS = O(n^3 + nlogn)$, $QLACS = O(n^3 + nlogn)$, $QLACS = On(n^2 + logn)$. The model is the range of

$$O(1) < On(n^2 + logn) < n^3 < e^3$$

In terms of the searching, the optimization even got better. The QL incorporates the ACS [101] for the searching to produce QLACS. The cost of searching is dependent on the use of shortest path by Dijkstra [29] (graph with $|V|$ vertices and $|E|$ edges) using a min-heap as priority queue. Based on Dijkstra algorithm for searching, the average and worst time complexity is $O((|V| + |E|)\log|V|)$ and the space complexity is $O((|V|)$. Since the ACS uses the Dijkstra and the QL adopts and depends on the ACS to do the searching. It therefore means that QL and ACS which is QLACS employ Dijkstra for searching. Thus the optimization cost of searching and communication give the following result.

	Time	Space
For ACS	$O((\|V\| + \|E\|)\log\|V\|)$	$O((\|V\|)$
For QL	↓ ↓	↓ ↓
For QLACS		

$$O(1) < O(log(|V| + |E|) < O(\log(n)) < O(n) < O(|V| + |E|)\log|V|) < O(|V|^2) < O(n^2)$$

3.7.4 Illustrations

In this section, some worked examples on cooperative safety inspections are explained using two robots. This thesis uniquely presents scenarios on the ease of implementation of the proposed QLACS. According to the explanation in section 3.7, the results obtained from the route-finding algorithm are displayed in Figure 3.12, as well as the parameters to be used in Figure 3.11. These results are used by the second component of our hybrid model to achieve cooperation and communication. The examples are explained in Figures 3.13 and 3.14.

Example I of QLACS (Using Figure 3.6 for optimized route finding)
Parameters used in the first component of QLACS are
$Q = 2.0, \alpha = 3, \beta = 2, \rho = 0.01$
Starting State: F
Terminating State: C/F
Terminating condition: When all states have been visited at least once and it is at terminating state
State Space = {F,G,E,D,A,B,C}
Initialize pheromones positions to 0.01
Rand = (0,1) returns a random value between 0 and 1
Equations
-Computation of attractiveness $\eta_{i,j}$ using equation 3.1
-Computation of instantaneous pheromones by ant k for all potential states using Equation 3.2
-Pheromone update using Equation 3.3
-Computation of probabilities for the potential states using Equation 3.4
-Adaptation of roulette wheel using Equation 3.5
Equation 3.4 can be reduced to Let $w(i,j) = [\tau_{i,j}]^{\alpha}\ [\eta_{i,j}]^{\beta}$
Sum $= \sum_{i=1}^{N+!} w(i,j)$
So $P_r(i,j) = \frac{w}{sum}$

Figure 3.11: Parameters for QLACS example I

Repeat steps 1 to 6 for subsequent current states until the termination condition and state are reached. At the end of seven updates we have Tables 3.8 and 3.9. The total length shown in table 3.8 represents the total number of obstacles encountered by each agent while trailing to achieve the optimal route for the robots. The number of obstacles shown as 10 is calculated using the trail result in Table 3.8. Table 3.9 displays the probability update table for a full cycle of route finding. In the case of this first example, the first robot will terminate its inspection through C then H.

Starting State: F

Potential States : D, E, G

1. Use Equation 3.1, $\eta_{F,D} = 1$, $\eta_{F,E} = 1$, $\eta_{F,G} = 1$
2. Use Equation 3.2, $L_k = 1, \Delta\tau^k(F,D) = \frac{2}{1} = 2$, $\Delta\tau^k(F,E) = 2$, $\Delta\tau^k(F,G) = 2$
3. Use Equation 3.3, $\tau_{F,D} = (1-0.01) * 0.01 + 2 = 2.0099 = 2.01, \tau_{F,E} = 2.01$, $\tau_{F,G} = 2.01$

First Pheromone Update

F	G	E	D	A	B	C
0.01	2.01	2.01	2.01	0.01	0.01	0.01

4. Use Equation 3.4 $w(F,D) = [\tau_{F,D}]^{\alpha}\,[\eta_{F,D}]^{\beta} = (2.01)^3 = 8.12, w(F,E) = 8.12, w(F,G) = 8.12$

Sum = $w(F,D) + w(F,E) + w(F,G) = 24.36$

$P_r(F,D) = \frac{w}{sum} = \frac{8.12}{24.36} = 0.33$, $P_r(F,E) = 0.33$, $P_r(F,G) = 0.33$

Probabilities

F	G	E	D	A	B	C
0	0.33	0.33	0.33	0	0	0

5. Use Equation 3.5

H	F	G	E	D	A	B	C
0	0	0.33	0.33	0.33	0	0	0

Call Rand

Rand = 0.47, Rand falls in state E. Roulette wheel selects E as the next state, end of roulette wheel.

6. Update trail: F, E

Current State: E

Potential States : B, D, G

1. Use Equation 3.1, $\eta_{E,B} = 1/2$, $\eta_{E,D} = 1/2$, $\eta_{E,G} = 1$
2. Use Equation 3.2, $L_k = (F - E - D) = 1 + 2 = 3$, $L_k = (F - E - B) = 1 + 2 = 3$, $L_k = (F - E - G) = 1 + 2 = 3$

$\Delta\tau^k = \frac{2}{3} = 0.67$

3. Use Equation 3.3, $\tau_{E,B} = (1-0.01) * 2.01 + 0.67 = 2.66, \tau_{E,D} = 2.66$, $\tau_{E,G} = 2.66$

Second Pheromone Update

F	G	E	D	A	B	C
0.01	2.66	2.01	2.66	0.01	2.66	0.01

4. Use Equation 3.4 $w(E,B) = (2.66)^3 * \left(\frac{1}{2}\right)^2 = 4.71$, $w(E,D) = 4.71, w(E,G) = (2.66)^3 * (1)^2 = 18.82$

Sum = $4.71 + 4.71 + 18.82 = 28.24$

$P_r(E,B) = \frac{w}{sum} = \frac{4.71}{28.24} = 0.17$, $P_r(E,D) = \frac{4.71}{28.24} = 0.17$, $P_r(E,G) = \frac{18.82}{28.24} = 0.67$

Probabilities

F	G	E	D	A	B	C
0	0.67	0	0.17	0	0.17	0

5. Use Equation 3.5

H	F	G	E	D	A	B	C
0	0	0.67	0	0.17	0	0.17	0

Call Rand

Rand = 0.7, Rand falls in state D. Roulette wheel selects D as the next state, end of roulette wheel.

6. Update trail: F, E, D

Figure 3.12: QLACS example I navigational analytical solution

Table 3.8: Pheromone update of a full cycle

Pheromone update								
Current states	F	G	E	D	A	B	C	
F	0.01	2.01	2.01	2.01	0.01	0.01	0.01	1st update
E	0.01	2.66	2.01	2.66	0.01	2.66	0.01	2nd pdate
D	3.13	0.01	3.03	0.01	3.03	0.01	0.01	3rd update
A	0.01	0.01	0.01	0.45	0.01	0.45	0.01	4th update
B	0.01	0.01	0.67	0.01	0.67	0.01	0.7	5th update
C	0.01	0.91	0.01	0.01	0.01	0.89	0.01	6th update
G	0.71	0.01	0.69	0.01	0.01	0.01	0.71	7th update
C = terminating state and ant k has moved through all states at least once Trail: FEDABCGC Number of obstacles = 1+2+2+2+1+1+1+1 = 10								

Table 3.9: Probability update of a full cycle

Probability table							
Current states	F	G	E	D	A	B	C
F	0	0.33	0.33	0.33	0	0	0
E	0	0.67	0	0.17	0	0.17	0
D	0.69	0	0.16	0	0.16	0	0
A	0	0	0	0.5	0	0.5	0
B	0	0	0.16	0	0.16	0	0.68
C	0	0.52	0	0	0	0.49	0
G	0.45	0	0	0	0	0	0.45
C = terminating state							

Figure 3.13 displays the parameters of the second example of QLACS. The second component of QLACS handles this aspect of the model, which is the communication and cooperative part. Once the first component hands the output to the second component, it becomes the second component input and it runs with it. From Figure 3.13, QLACS_R1 represents the input for R1 and QLACS_R2 represents the input for R2 received from the first component of QLACS. The terminating condition is when all states have been visited.

Example II of QLACS (For good cooperation and communication between robots)
Parameters used in the second component of QLACS (Using output from the first component of QLACS)
Reward Scheme: Inspect = 50, Ignore = 100, Shutdown = 150
State space: Optimized path from QLACS_R1 = {F,E,D,A,B,C,G,C} and QLACS_R2 = {C,B,A,D,F,E,G,C}
Starting State: F/C
S_j = Terminating State: C then H
Terminating condition: when all states have been visited.
Initialize Qvalue positions to zeros
Equations
-Compute learning rate using Equation 3.8
-Compute update on $Q(s, a)$ using Equation 3.9
-Do broadcast (share intelligence) using Equation 3.10

Figure 3.13: Parameters for QLACS example II

The rest of example II displayed in Figure 3.14 explains the cooperative behavioural scenario from one state to another for two robots. The tables in the last row of Figure 3.14 show the communication and cooperative output achieved using the second component of QLACS.

Starting Simulation Robot 1
Starting State: F

1. Use equation 3.8
   $\gamma(F) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
2. Check the Qvalue for state F (Use equation 4.10)
   $Q(F, a) = 0$
   Selected action, a = inspect
3. Use equation 3.9
   $Q(F, a) = 50 + 0.25(0) = 50$
   End of value iteration

Current State: E, Robot 1

1. Use equation 3.8
   $\gamma(E) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
2. Check the Qvalue for state E (Use equation 4.10)
   $Q(E, a) = 0$
   Selected action, a = inspect
3. Use equation 3.9
   $Q(E, a) = 50 + 0.25(\max(0,0,0) = 50$
   End of value iteration

Current State: C, Robot 2

1. Use equation 3.8
   $\gamma(C) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
2. Check the Qvalue for state C (Use equation 4.10)
   $Q(C, a) = 0$
   Selected action, a = inspect
3. Use equation 3.9
   $Q(C, a) = 50 + 0.25(\max(0,0,0) = 50$
   End of value iteration

Current State: B, Robot 2

1. Use equation 3.8
   $\gamma(B) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
2. Check the Qvalue for state B (Use equation 4.10)
   $Q(B, a) = 0$
   Selected action, a = inspect
3. Use equation 3.9
   $Q(B, a) = 50 + 0.25(\max(0,0,0) = 50$
   End of value iteration

Current State: D, Robot 1

1. Use equation 3.8
   $\gamma(D) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
2. Broadcast (Use equation 4.10)
   $Q(D, a) = 0$
   Selected action, a = inspect
3. Use equation 3.9
   $Q(D, a) = 50 + 0.25(\max(0,0,0) = 50$
   End of value iteration

Current State: A, Robot 2

1. Use equation 3.8
   $\gamma(A) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
2. Broadcast (Use equation 4.10)
   $Q(A, a) = 0$
   Selected action, a = inspect
3. Use equation 3.9
   $Q(A, a) = 50 + 0.25(\max(0,0,0) = 50$
   End of value iteration

Current State: A, Robot 1

1. Use equation 3.8
   $\gamma(A) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$
2. Broadcast (Use equation 4.10)
   $Q(A, a) = 50$, i.e. $Q > 0$
   Selected action, a = Ignore
3. Use equation 3.9
   $Q(A, a) = 100 + 0.4(\max(0,0,0) = 100$
   End of value iteration

Current State: D, Robot 2

1. Use equation 3.8
   $\gamma(D) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$
2. Broadcast (Use equation 4.10)
   $Q(D, a) = 50$ I,e. $Q > 0$
   Selected action, a = ignore
3. Use equation 3.9
   $Q(D, a) = 100 + 0.4(\max(0,0,0) = 100$
   End of value iteration

Current State: B, Robot 1

1. Use equation 3.8
   $\gamma(B) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$
2. Broadcast (Use equation 4.10)
   $Q(B, a) = 50$, i.e. $Q > 0$
   Selected action, a = Ignore
3. Use equation 3.9
   $Q(B, a) = 100 + 0.4(\max(0,0,0) = 100$
   End of value iteration

Current State: F, Robot 2

1. Use equation 3.8
   $\gamma(D) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$
2. Broadcast (Use equation 4.10)
   $Q(F, a) = 50$ I,e. $Q > 0$
   Selected action, a = ignore
3. Use equation 3.9
   $Q(F, a) = 100 + 0.4(\max(0,0,0) = 100$
   End of value iteration

Current State: C, Robot 1

1. Use equation 3.8
   $\gamma(B) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$
2. Broadcast (Use equation 4.10)
   $Q(B, a) = 50$, i.e. $Q > 0$
   Selected action, a = Ignore
3. Use equation 3.9
   $Q(B, a) = 100 + 0.4(\max(0,0,0) = 100$
   End of value iteration

Current State: E, Robot 2

1. Use equation 3.8
   $\gamma(E) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$
2. Broadcast (Use equation 4.10)
   $Q(E, a) = 50$ I,e. $Q > 0$
   Selected action, a = ignore
3. Use equation 3.9
   $Q(E, a) = 100 + 0.4(\max(0,0,0) = 100$
   End of value iteration

Current State: G, Robot 1

1. Use equation 3.8
   $\gamma(G) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
2. Broadcast (Use equation 4.10)
   $Q(G, a) = 0$
   Selected action, a = Inspect
3. Use equation 3.9
   $Q(G, a) = 50 + 0.25(\max(0,0,0) = 50$
   End of value iteration

Current State: G, Robot 2

1. Use equation 3.8
   $\gamma(G) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$
2. Broadcast (Use equation 4.10)
   $Q(G, a) = 50$ I,e. $Q > 0$
   Selected action, a = ignore
3. Use equation 3.9
   $Q(G, a) = 100 + 0.4(\max(50,50,50) = 100 + 20 = 120$
   End of value iteration

Current State: C, Robot 1

1. Use equation 3.8
   $\gamma(C) = \frac{0.5}{[1+0.4]} = \frac{0.5}{1.4} = 0.36$
2. Broadcast (Use equation 4.10)
   $Q(C, a) = 50$
   Selected action, a = Ignore
3. Use equation 3.9
   $Q(C, a) = 100 + 0.36(\max(50,50) = 100 + 18 = 118$
   End of value iteration

Current State: C, Robot 2

1. Use equation 3.8
   $\gamma(C) = \frac{0.5}{[1+0.36]} = \frac{0.5}{1.36} = 0.37$
2. Broadcast (Use equation 4.10)
   $Q(C, a) = 50$ I,e. $Q > 0$
   Selected action, a = ignore

All QLACS1 states exhausted
Goal State: H, Robot 1

1. Use equation 3.8
   $\gamma(H) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
2. Broadcast (Use equation 4.10)
   $Q(H, a) = 0$
   Selected action, a = Shutdown
3. Use equation 3.9

All QLACS2 states exhausted
Goal State: H, Robot 2

4. Use equation 3.8
   $\gamma(H) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$
5. Broadcast (Use equation 4.10)
   $Q(H, a) = 0$
   Selected action, a = Shutdown
6. Use equation 3.9

<table>
<tr>
<td>3. Use equation 3.9
$Q(C,a)$
$= 100 + 0.37(\max(50,50)$
$= 100 + 0.37 * 50 = 118.5$
End of value iteration</td>
<td>$Q(H,a) = 150 + 0.25(\max(0,0)$
$= 150$
End of value iteration</td>
<td>$Q(H,a) = 150 + 0.25(\max(0,0)$
$= 150$
End of value iteration</td>
</tr>
<tr>
<td>End of policy iteration</td>
<td>Robot 1
<table>
<tr><th></th><th>Inspect</th><th>Ignore</th><th>Shutdown</th></tr>
<tr><td>F</td><td>Yes</td><td>No</td><td>No</td></tr>
<tr><td>E</td><td>Yes</td><td>No</td><td>No</td></tr>
<tr><td>D</td><td>Yes</td><td>No</td><td>No</td></tr>
<tr><td>A</td><td>No</td><td>Yes</td><td>No</td></tr>
<tr><td>B</td><td>No</td><td>Yes</td><td>No</td></tr>
<tr><td>C</td><td>No</td><td>Yes</td><td>No</td></tr>
<tr><td>G</td><td>Yes</td><td>No</td><td>No</td></tr>
<tr><td>C</td><td>No</td><td>Yes</td><td>Yes through H</td></tr>
</table></td>
<td>Robot 2
<table>
<tr><th></th><th>Inspect</th><th>Ignore</th><th>Shutdown</th></tr>
<tr><td>C</td><td>Yes</td><td>No</td><td>No</td></tr>
<tr><td>B</td><td>Yes</td><td>No</td><td>no</td></tr>
<tr><td>A</td><td>Yes</td><td>No</td><td>No</td></tr>
<tr><td>D</td><td>No</td><td>Yes</td><td>No</td></tr>
<tr><td>F</td><td>No</td><td>Yes</td><td>No</td></tr>
<tr><td>E</td><td>No</td><td>Yes</td><td>No</td></tr>
<tr><td>G</td><td>No</td><td>Yes</td><td>no</td></tr>
<tr><td>C</td><td>No</td><td>Yes</td><td>Yes through H</td></tr>
</table></td>
</tr>
</table>

Figure 3.14: QLACS example II cooperative behaviour

3.8 Chapter Summary

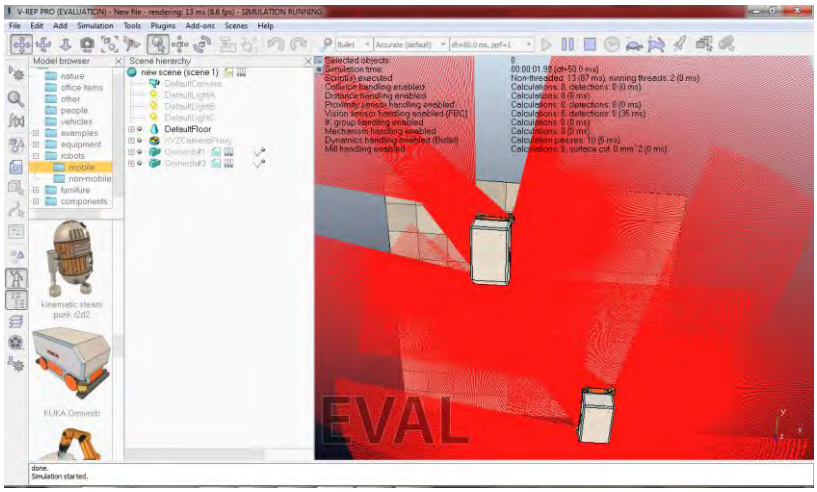
In this chapter, the proposed research problem was re-established and the research methodology was discussed in detail. The problem was formulated for the proposed model. The methods and functions for the model were outlined and explained. The algorithms for the new model were discussed and the processes involved in formulating the algorithms were also explained. Analytical formulation of the problem was presented and worked examples were done.

4. Experimental Evaluations of QLACS

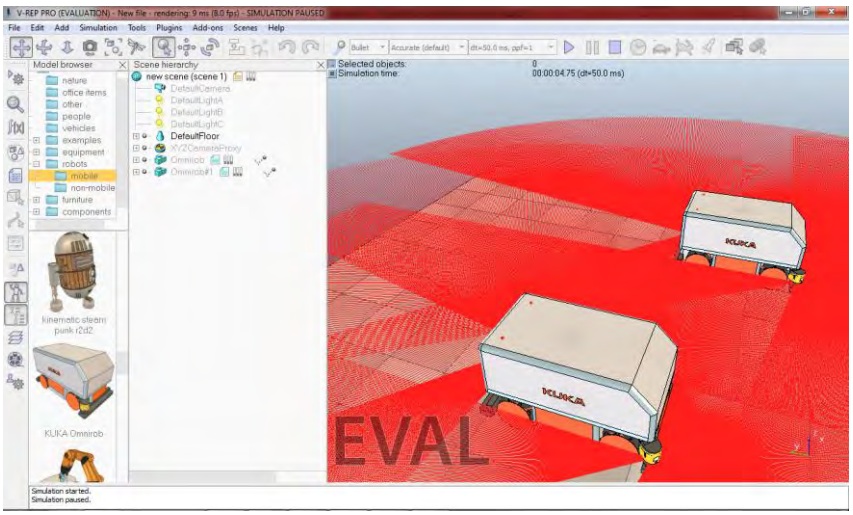
To establish the effectiveness of QLACS for cooperative behaviour in the underground terrains, some simulation experiments coded in C# are conducted. The experimental results of implementing QLACS in an environment that consists of obstacles and links are tabulated in this section. The experimental setup is explained in 4.1. Different performance categories are shown in this section: without communication category and with communication category. In the without communication category, as displayed in section 4.2, we found that robots can inspect all states individually without knowing that another robot exists. Robots can also inspect some of the states, thereby leaving some states not inspected. The communication category is explained in section 4.3 while the performance of the QLACS measured with other existing methods is tabulated in sections 4.5 and 4.6.

4.1 Experimental Setup

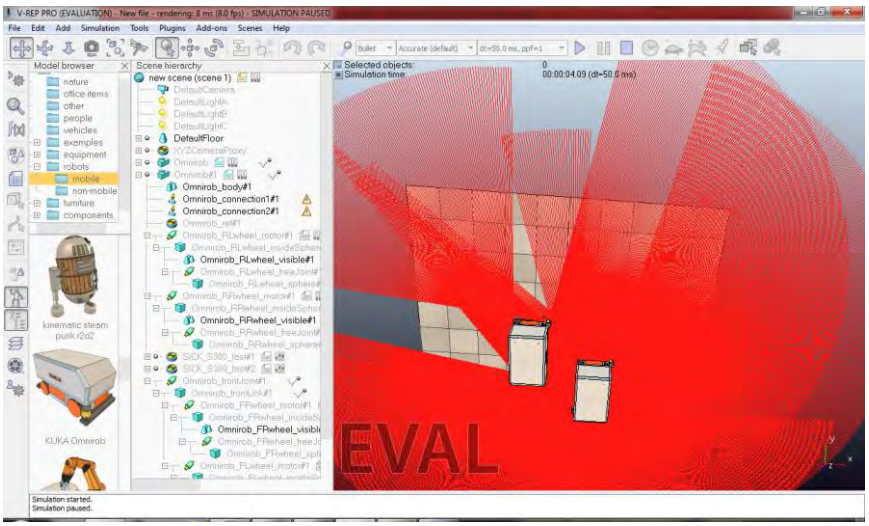
Figure 4.1 displays different sets of experiments conducted in the environment using two robots. Figure 4.1(a) shows how two robots resume inspection from two different entrances. In each set of experiments, the robots take the readings of the roof cracks and level of TG using their sensors. The same behaviours happen in Figures 4.1(b) and 4.1(c) and 4.1(d) respectively at different inspection locations. The inspection locations vary in the four experimental setups shown in Figure 4.1.



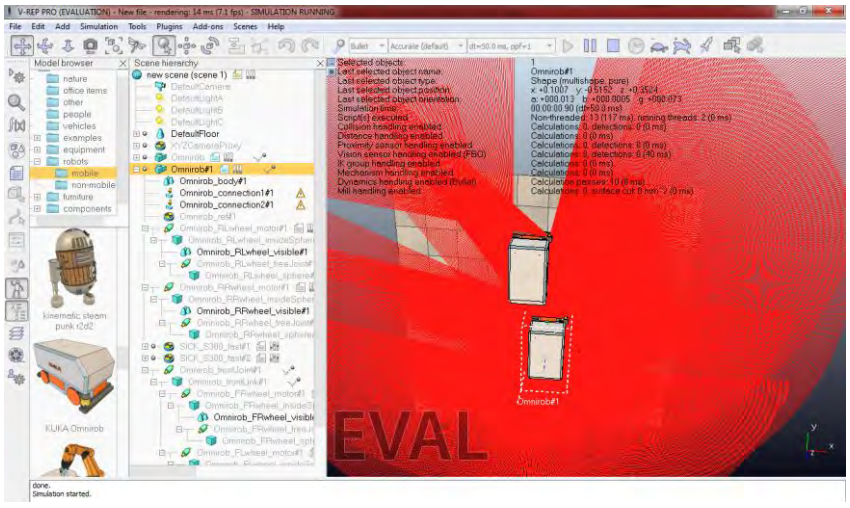
(a) Two robots starting inspection from two different entrances



(b) Two robots inspecting as they navigate in the environment



(c) Two robots inspecting different locations



(d) Another inspecting position for two robots

Figure 4.1: Different experimental behaviours for two robots

4.2 Experiment 1: Performance of QLACS without Cooperation

The result of implementing the QLACS without communication in the proposed environment (Figures 1.5(page 7) and 3.4 (page 48)) is shown in Tables 4.1 and 4.2. In the case of Table 4.1, R1, enters the mine through state F while R2 enters the mine through state C. However, each robot learns by inspecting some of the states and ignoring some of the states. Since there is no communication, they exit the mine after learning, consequently not inspecting all the states. The same routine is reflected in Table 4.2, but in this case, each robot ends up inspecting all the states before exiting the mine. Analysis of Tables 4.1 and 4.2 shows that resources are wasted and the inspection job is not effective and efficient. Comparing the two tables, the time and distance costs are high, though higher in Table 4.2 because of many repetitions. For instance, the time and distance cost in Table 4.2 column 2 are 48.0028 and ((F,G,E,D,A,B,C), (C,B,A,D,E,G,F)) respectively. It also shows that the states are repeated in Tables 4.1 and 4.2. The memory usage is equally high. This led us to create a more efficient QLACS with communication by introducing some heuristics. The processes involved in Tables 4.1 and 4.2 are explained in Figure 4.2.

Table 4.1: QLACS without communication (inspecting Some States)

No of run	1	2	3	4	5	6	7
Iterations	9	10	10	13	13	9	13
Time (secs)	43.0025	30.0017	34.002	30.0017	31.0017	31.0018	27.0016
Memory usage(bytes)	18872	18160	19204	18208	17896	18164	18308
Inspected states (R1)	G	F,G,C	C	F,D,B	E,A	F,E,D,A,C	F,G,E,A
Inspected states (R2)	B,A,G	C	C,B,A,E,F	B,A	C,B,E	B,A,E	A,G

Table 4.2: QLACS without communication (inspecting all states)

No of run	1	2	3	4	5	6	7
Iterations	114	10	70	10	9	10	10
Time (secs)	43.0024	48.0028	41.0023	34.0019	34.0019	34.0019	35.002
Memory usage(bytes)	28440	17960	27216	17000	18456	17672	17968
Inspected states (R1)	F,G,E,D,A,B,C	F,G,E,D,A,B,C	F,G,E,D,A,B,C	F,G,E,D,A,B,C	F,G,E,D,A,B,C	F,G,E,D,A,B,C	F,G,E,D,A,B,C
Inspected states (R2)	C,B,A,D,E,G,F	C,B,A,D,E,G,F	C,B,A,D,E,G,F	C,B,A,D,E,G,F	C,B,A,D,E,G,F	C,B,A,D,E,G,F	C,B,A,D,E,G,F

One can see from Tables 4.1 and 4.2 that there is no good communication among the robots, hence the evolvement of Table 4.3.

(a) For Table 4.1	(b) For Table 4.2
Inspecting some States 1. Initialize the starting and goal states 2. Initialize all Q-values to zeroes 3. Select a random action If Q-value of all possible actions at current state are zeroes 4. Select the action with the highest Q-value If it is the Max Q-value 5. Compute and update Q-value of the selected action 6. Get new state among possible states 7. If **new state= goal state** then go to **steps 5 and 9** 8. Repeat **steps 3 to 6** until **step 7** 9. Shutdown	**Inspecting all States** 1. Initialize the starting and goal states 2. Initialize all Q-values to zeroes 3. Select a random action If Q-values of all possible actions at current state are zeroes 4. Select the action with the highest Q-value If it is the Max Q-value 5. Compute and update Q-value of the selected action 6. Get new state among possible states 7. If **new state= goal** state then go to **step 5 and 10** 8. Repeat **steps 3 to 7** until **step 9** 9. All states except the goal state has taken **Action= Inspect** 10. Shutdown

Figure 4.2: Processes used in achieving Tables 4.1 and 4.2

4.3 Experiment 2: Performance of QLACS with Good Cooperation

The heuristic added to the known QL made this experiment show good communication. This is where our contribution to communication is shown. As a robot inspects and learns the environment, it broadcasts its Q-values to the other robot in the form of a lookup table. In this case, each robot checks for Q-values; when a Q-value is equal to zero, the robot randomly selects a state for inspection. If a Q-value is greater than zero, the robot checks if the state is available for inspection or for ignoring. When a robot encounters a state with a Q-value equal to zero and the thread next to the state is equal to the goal state (H), then it shuts down. It must have checked the lookup table to see that all states have been inspected. The results in Table 4.3 show good communication between two robots. No states were inspected more than once. The iterations for every run with their times, memory usage and effective communication are also displayed in Table 4.3.

Table 4.3: QLACS with communication

No of run	1	2	3	4	5	6	7
Iterations	9	10	9	13	10	10	9
Time (secs)	33.0019	31.0017	31.0018	31.0018	29.0023	30.0017	31.0018
Memory Usage (bytes)	15748	15948	15748	18232	16576	15948	15748
Inspected states (R1)	F,G,E	F,G,E,D	F,G,E	F,E	F,G,E,D,B	F,G,E,D	F,G,E
Inspected states (R2)	C,B,A,D	C,B,A	C,B,A,D	F,G,E,D,A	C,A	C,B,A	C,B,A,D

Comparing Table 4.3 with Tables 4.1 and 4.2, one cannot but notice the huge differences in the memory, time and distance costs. The communication between the robots in Table 4.3 resulted in achieving good inspection; however, the random method of choosing next inspection states did not give an optimized route, thereby increasing time cost in passing and checking through inspected states.

4.4 Experiment 3: Performance of QLACS for the Navigation Behaviour of the Proposed Model

The result of implementing the first component of QLACS for effective navigation of the proposed model is tabulated in Table 4.5. Table 4.4 shows the selected parameters used in achieving the optimal path. The optimal path found after nine test runs is the path with a distance cost of 10 for both entries to the environment, displayed in Table 4.5. Therefore, columns 4, 5, 7, 8 and 9 can be used as the optimized path input for the first component of QLACS. The QLACS will use any of the optimized paths to navigate to the specified states and take decisions accordingly. For instance, the test run result for nine ants gives 16 iterations under 60.0035 seconds and the shortest paths to both robots coming from entries F and C of the environment. The path that gives the cost as 10 is obtained from FEDABCGC (1.2.2.2.1.1.1) and CBADFGEGC (1.2.2.1.1.2.2.1).

Alpha (α), Beta (β) and Rho (ρ) represent the heuristic properties of the ACS algorithm. The Alpha factor is the measure of the influence of pheromone concentration that can influence the selection of the path with the highest pheromone concentration. Beta is the measure of the influence that is related to the distance between any two adjacent nodes. It is also a heuristic factor that can measure the influence distance in selecting the next state. It is not limited to pheromones. Rho has to do with the rate at which the pheromones evaporate. Rho shows how often new paths are explored by the agents/ants rather than reinforcing old paths. Each agent cooperates by having access to other agents' pheromone values. The pheromone value is initialized to 0.01 and it is continually updated until learning stops. It is similar to the first component of QLACS, where we initialize all QLACS positions to zero and update for every new step.

Table 4.4: Navigation behaviour parameter specification

ACO Properties	Type of ACO	Population	Length of path	Pheromone coefficient β	Heuristic coefficient α	Evaporation rate ρ
Properties	ACS	9 or 12	8	2	3	0.01

Table 4.5: Navigation behaviour computations

No of Runs	1	2	3	4	5	6	7	8	9
Number of Ants	5	7	8	9	10	11	12	15	28
Iterations	12	26	14	16	23	14	16	23	22
Time (secs)	37.0021	37.0021	39.0022	60.0035	32.0019	33.0019	35.002	43.0025	65.0037
Best Train Distance (R1)	12	12	10	10	10	10	10	10	10
Best Trail Distance (R2)	10	12	12	10	10	12	10	10	10

Table 4.5 gave the optimized route to be used by the robots to achieve inspection tasks. This resulted in combining Tables 4.3 and 4.5 to achieve Table 4.6. Table 4.6 shows optimized time cost, memory usage, route cost and good communication.

4.5 Experiment 4: Benchmarking the New Hybrid Model (QLACS) with Popular Methods

Choosing the optimized paths, QLACS performs cooperative inspection behaviour through communication among robots. Looking at the sample result on Table 4.6, QLACS chooses the shortest and complete possible inspection routes from different runs. In this case, the best paths were obtained from Table 4.5 by using 9 agents, 10 agents and 12 agents. All the test runs gave best trail paths from both entrances listing the complete states with length 10, as shown in Table 4.5, i.e., they have states from F to C and from C to F. The length is the addition of weights, along the line (trail edges). Then the QLACS uses the optimized paths to make decisions on inspections. The result from Table 4.6 shows that no state is inspected twice or visited more than required. The QLACS model concentrates on making the best inspection decisions for MRS. The first run on Table 4.5 shows that nine agents/ants were used for the route finding, the optimized route was achieved after nine iterations under 7.0004 seconds and the states where all inspected effectively without redundancies.

Table 4.6: New hybrid model QLACS computations

No of Runs	1	2	3	4	5	6	7	8	9
Iterations	9	9	9	9	9	9	9	9	9
Number of Ants	9	9	9	10	10	10	12	12	12
Time (secs)	7.0004	8.0005	9.0005	11.0006	11.0006	12.0007	8.0005	9.0005	9.0006
Memory Usage(bytes)	10288	10280	10248	10288	10288	10288	10164	10712	10164
Inspected states (R1)	F,G,E,D	F,G,E,D	F,G,E,D	F,G,E,A,B	F,G,E,A,B	F,G,E,A, B	F,G,D,A	F,G,D,A	F,G,D,A
Inspected states (R2)	C,B,A	C,B,A	C,B,A	C,D	C,D	C,D	C,B,E	C,B,E	C,B,E

4.6 Experiment 4: Benchmarking the New Hybrid Model (QLACS) with Popular Methods

Table 4.7: Time cost comparison for QL, ACS and QLACS

Runs	QL Time Cost(secs)	ACS Time Cost(secs)	QLACS Time Cost(secs)
1	33.0019	37.0021	7.0004
2	31.0019	37.0021	8.004
3	31.0018	39.0022	9.0005
4	31.0018	32.0019	9.0005
5	29.0023	35.002	11.0006
6	30.0017	43.0025	12.0007
7	31.0018	60.0035	9.0006
Average	30.8590	40.4309	9.2868

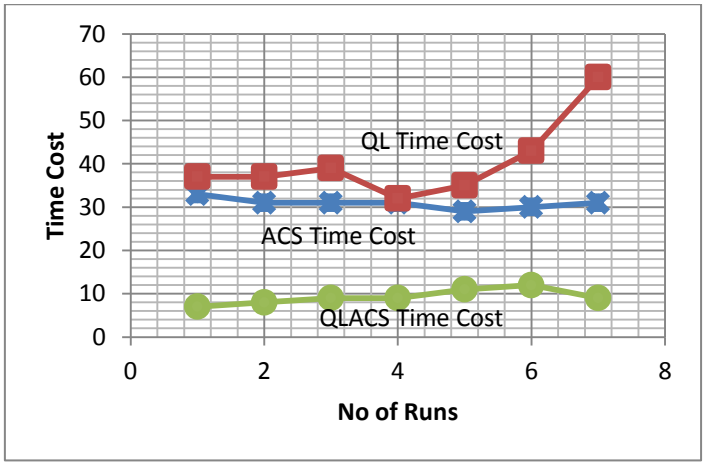


Figure 4.3: Comparison of time costs for QL, ACS and QLACS

Based on the results tabulated in Table 4.7 and Figure 4.3, the average time cost of achieving the MRS behaviours for QL, ACS and QLACS were compared. Two robots will use an average of 30.8590 secs to achieve thorough inspection behaviour using QL, an average of 40.4309 seconds to achieve optimal navigation behaviour using ACS and an average of 9.2868 seconds to achieve both

navigation and cooperative behaviour using QLACS. The result shows that our proposed integrated algorithm performs better with reduced time cost. On the same note, the route costs for the QL and QLACS were also compared. The results in Table 4.8 and Figure 4.4 show that the proposed model QLACS gave a much lower route cost than the QL.

Table 4.8: Route cost comparison for QL and QLACS

	QL (secs)		**QLACS (secs)**	
Runs	Path cost for R1	Path Cost for R2	Path cost for R1	Path cost for R2
1	20	20	10	10
2	32	19	10	10
3	28	14	10	10
4	27	27	10	10
5	39	30	10	10
Average	29.5	22	10	10

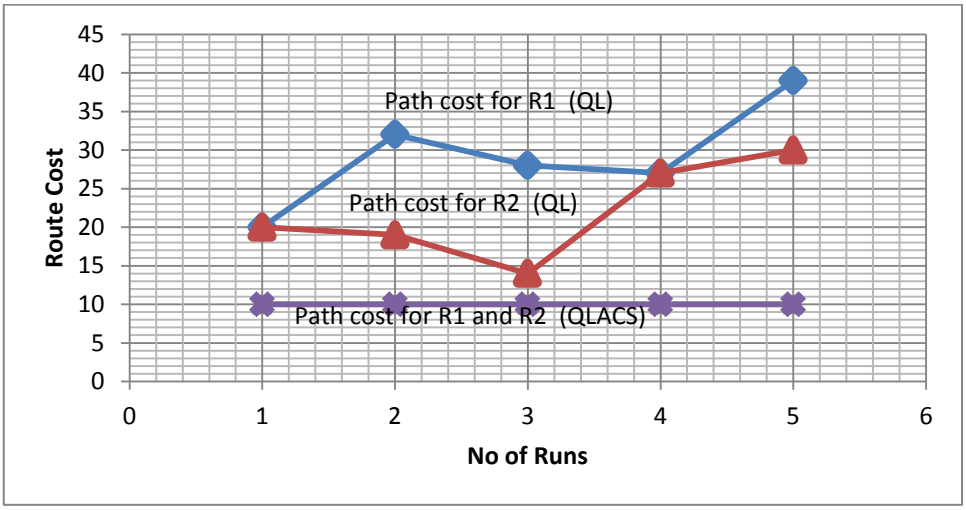


Figure 4.4: Comparison of route costs for QL and QLACS

The number of ants and iterations used to achieve cost-effective routes are displayed in Figure 4.5. The more ants, the more iterations. The best routes created in the five test runs shown in Figure 4.5 are run numbers 1 and 3. They used fewer ants and iterations to achieve the optimal routes. The optimal result for the proposed model is achieved under nine iterations for every run. The iterations remain constant for any number of agents and robots. The blue line with star markers in Figure 4.6 is the iteration value for 9 runs. The red line shows the different amounts of time required for each run. The time is also almost stable for the QLACS, though it fluctuated a little in the middle. This

signifies that the proposed model is more robust and effective in finding the optimal route and coordinating MRS. The reduced route cost and shorter computation time achieved with the QLACS satisfied the criteria for cooperative behavioural purposes.

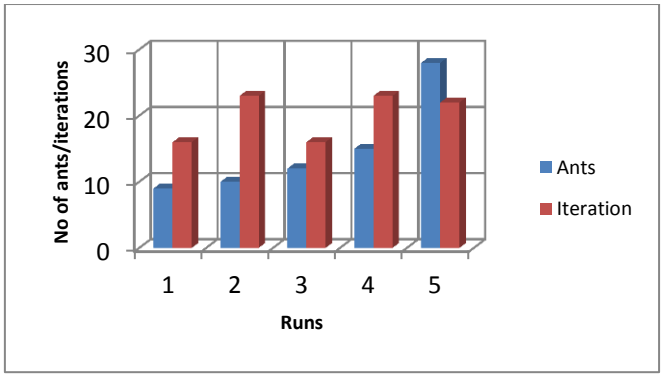


Figure 4.5: Number of ants/iterations required for QLACS to find optimal route

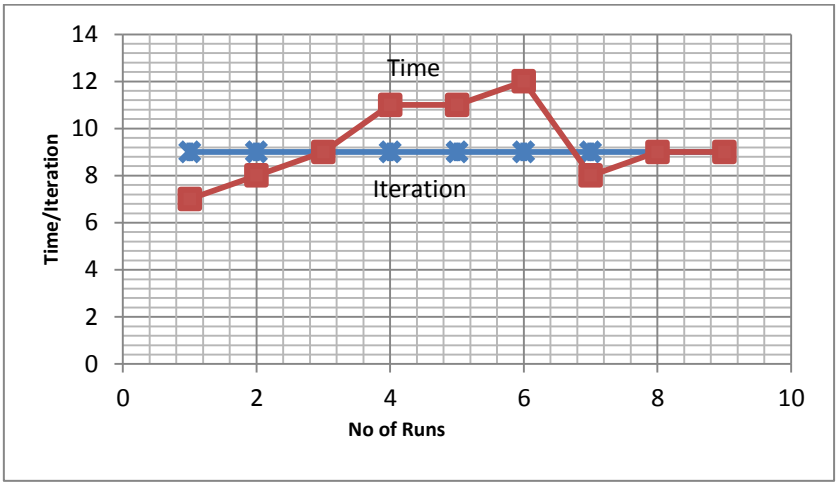


Figure 4.6: Time and iterations required for QLACS to achieve optimal behaviour

4.7 Chapter Summary

The formulated problem was simulated and the results were tabulated and explained this chapter. The cooperative behavioural model was experimented and some performance evaluations were carried out. The results of the experiments show that the model is promising.

5. Scalability Analysis of QLACS

Qualitative and quantitative experimental evaluation approaches are considered in this section. The main focus of this work is using multi-robots to achieve cooperative inspection tasks in underground terrains. The implementation designs and the class diagrams showing interactions of objects are explained in 5.1. For the experiments conducted in this work, different tests of the developed model are performed in 5.2, 5.3 and 5.4 using two, three and four robots respectively. The experiments on time and memory scalability observation are performed in 5.5, while the comparative analysis of cooperative behaviour systems is detailed in 5.6.

5.1 Implementation Design

In this section, our algorithm, which is realized as a program, is illustrated in Figures 5.1 and 5.2. The scenario in Figure 5.1 shows the underground terrain with two communicating robots exhibiting inspection behaviour. The environment is partitioned into seven different rooms. The first robot, R1, takes action from state F to state E and communicates its action to R2. R2 then takes action from state C to state B and communicates to R1. This process is continued until the environment has been fully inspected and each robot exits the terrain via the closest exit point to H.

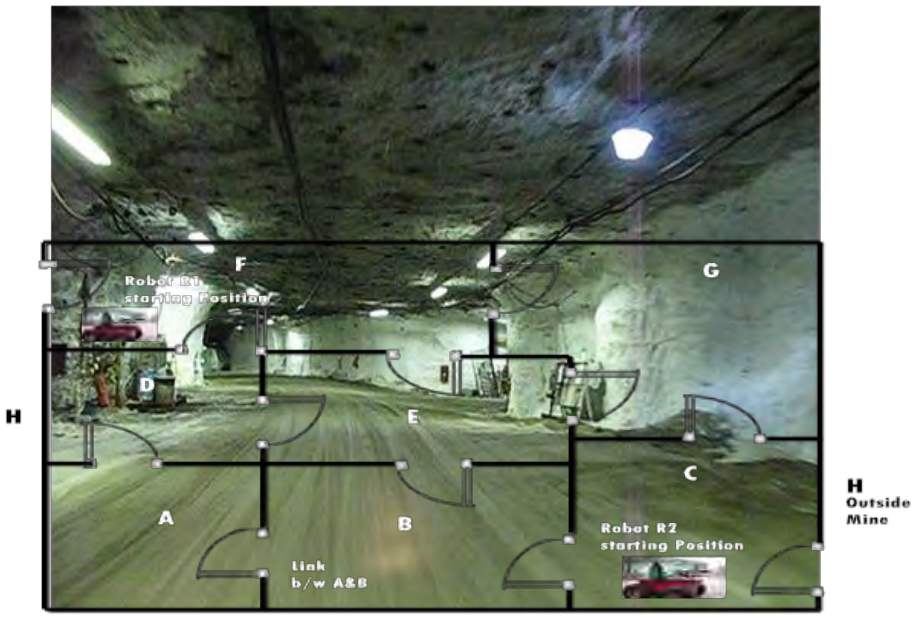


Figure 5.1: Underground terrain with segmented regions and two robots

The proposed model for the cooperating behaviour of autonomous robots has five integrating classes: the ACS, thread library, sensor, QL and graph classes. The ACS class does the route finding and it is the genesis class that must provide two optimized paths for the robots coming from the two entrances or exits of the mines. We refer to this model of the environment as two-exit and entrance (2EE). These optimized routes are relayed to the QL class during inspection. The QL class has a direct association with the thread library class. The thread library class is also associated with the ACS class. The QL class is supposed to make the robots learn which decision is best when the sensor class renders environmental readings of the mine. The sensor readings could indicate a crack in the roof of the mine (RC) or level of gaseous toxicity in the mine environment (TG). The sensor class manages the range of values as perceived from the sensors on the autonomous robots. The decisions of the robots are determined by the modified accelerated QL algorithm, which could be [Inspect], [Ignore] or [Shutdown]. The QL class communicates with the sensor class while the agent is passing through each position in the mine, because the inspection must be complete and thorough. The graph class is the last class that picks up accumulated values from the learning class and populates the chart control based on the customized context of parameters we want to use in measuring the overall performance. Figure 5.2 is the unified modelling language pictorial view of the classes.

It shows the key building blocks that enable designers to represent a given system's classes, the attributes and functions that are associated with them and the relationship among the different classes that make up the system. Both ACS and QL are associated directly with the thread library. We rely on the thread library (task factory) in Microsoft DotNet library to create several instances of threads in parallel, may-be one to N number of threads. We used this thread library because from the available documentation it is adjudged the best and provides a thread safe environment. It also has the advantage of dealing with delegates and methods that return value, unlike other libraries that do not allow values to be returned. This makes it easier to access and manipulate individual Q-values and sensor readings in its own thread without race conditions or deadlock and also makes the thread library thread safe.

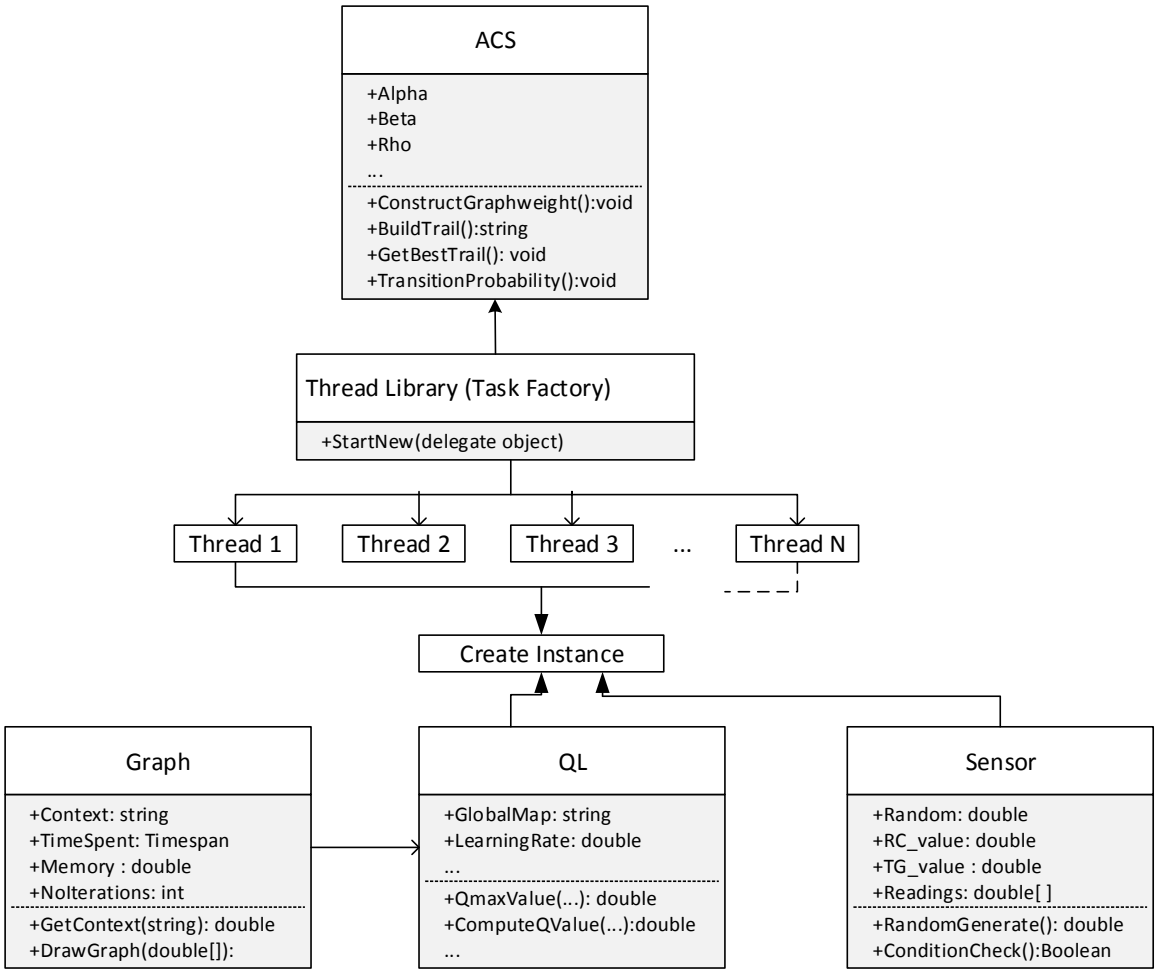


Figure 5.2: Class diagrams showing interactions of objects

Each thread creates an instance of the QL class and sensor class. The sensor class simulates or manages readings from the hardware devices. The sensor class is to detect two conditions: roof crack (RC_Value) and toxic gas level (TG_Value). The QL class learns based on these values to determine what actions are to be carried out. The QL class supports communications among the competing threads, i.e. they share intelligence about states they had visited. Through this communication the threads are aware of other robots, thereby avoiding multiple inspections with reduced inspection time.

Through the capability in the thread library, each thread has access to the optimal path (from ACS). When a thread gets to a state, it calls the sensor class to deliver its readings to the QL class then

decisions are taken. The QL class uses the *Display* method to show all results of learning by each thread.

The graph class relies on the parameters supplied and Q-values generated during the exercise to plot a chart. The charts are based on the selected context of performance. The context of performance refers to the performance yardsticks, such as memory usage, inspection time, states inspected by each robot against the number of robots etc.

The notations on Figure 5.2 have the following meaning:

- At least one sensor is needed for one robot to sense the environment.
- At least one robot with QL thread is needed to do a complete inspection.
- The graph gives details of at least one learning robot.

In summary, we use at least two robots with two QL threads to achieve our cooperative behavioural model development.

5.2 Experiment 1: Performance of QLACS Using Two Robots

In this section, we present some results obtained by experimenting with the cooperative behavioural action of two robots using the QLACS model. The performance of the two robots was evaluated by running the simulation 10 times, using the same number of agents. The performance of the QLACS model shows good communication between the two robots under the states inspected, in the last two columns of Table 5.1. The time and memory used in achieving these inspections are recorded in columns 2 and 3 of Table 5.1. The time for inspection for two robots is between the range of 7 – 11 and the memory usage is within the same range. Having achieved good cooperation between two robots, Figure 5.3 depicts the simulated movement of the two robots. The blue long broken lines depict the movement of R1, while the red short broken lines depict R2's movement. The blue double arrows show the states inspected by R1 and the red double arrow lines show the states inspected by R2. The flow of movement and the states visited by each robot according to Table 5.1 and Figure

5.3 create accurate understanding of the cooperation between the two robots. The route followed by R1 is FEDABCGC and the route followed by R2 is CBADFEGC.

Table 5.1: Performance of QLACS with two robots-based MRS

No of runs	Time (sec)	Memory usage (bytes)	Inspected states (R1)	Inspected states (R2)
1	7.0004	10304	F,G,E,D	C,B,A
2	10.0006	10300	F,G,E,D	C,B,A
3	10.0006	10264	F,G,E,D	C,B,A
4	11.0007	10296	F,G,E,D	C,B,A
5	11.0006	10296	F,G,E,D	C,B,A
6	8.0005	10288	F,G,E,D	C,B,A
7	7.0004	10288	F,G,E,D	C,B,A
8	7.0004	10256	F,G,E,D	C,B,A
9	7.0004	10836	F,G,E,D	C,B,A
10	8.0004	10288	F,G,E,D	C,B,A

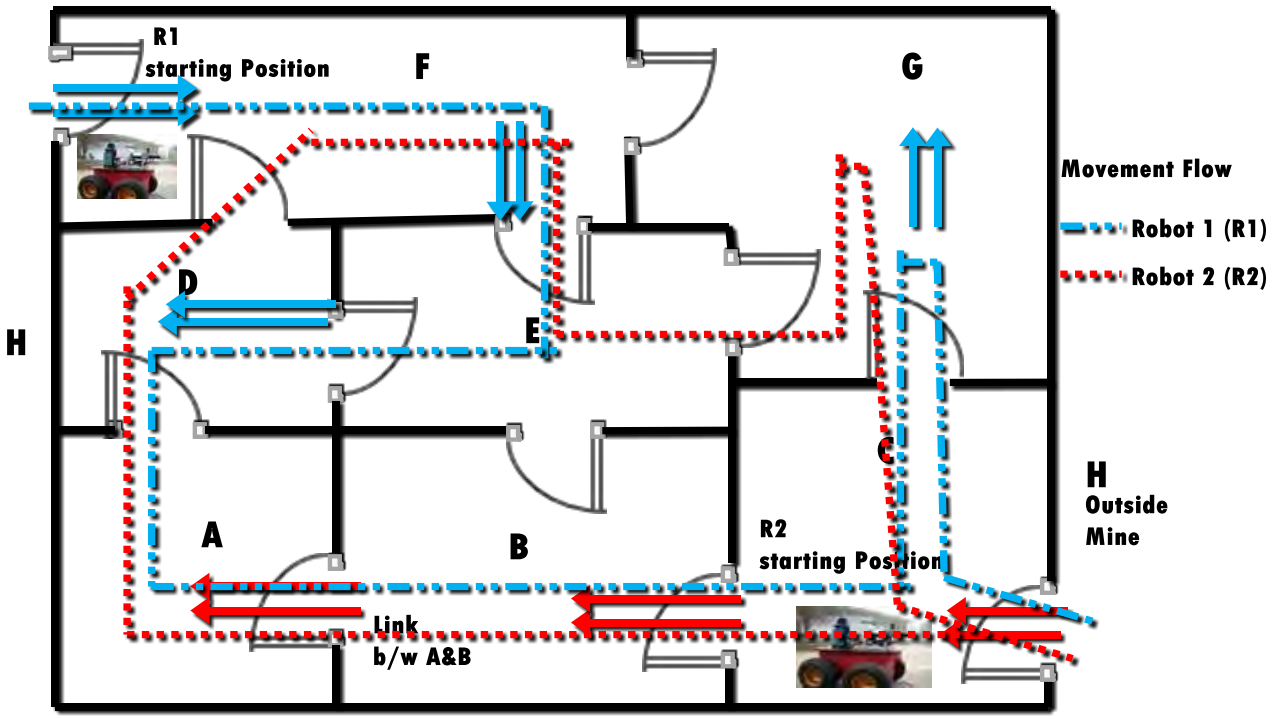


Figure 5.3: Movement of two robots showing cooperation

5.3 Experiment 2: Performance of QLACS Using Three Robots

In investigating the scalability of the QLACS model, three robots are used in an experiment on cooperative inspection of the underground terrain. The simulated environment is run 10 times just as when using the two robots. The performance obtained from the team of three robots was compared to the performance from the team of two robots; although there is slight increase in time and memory

usage for three robots, the three robots achieve good communication and effective inspection, as shown in Table 5.2, because there are no repetitive inspections. The flow of movement and the states inspected by each robot are depicted in the simulated movement of the three robots in Figure 5.4. For the entry points of the three robots, as shown in Figure 5.4, we use the result obtained from the optimized route-finding component of the QLACS model. R1 enters the terrain through state F and exits through state C, R2 and R3 enter through state C and exit through state C. The flow of movement for the three robots as shown in Figure 5.4 are, R1: CBADFEGCH, R2: FEDABCGCH AND R3: CBADFEGCH. Results show that QLACS can handle three cooperating robots conveniently with an average time of 12.7 and almost stable memory usage.

Table 5.2: Performance of QLACS with three robots-based MRS

No of runs	Time (sec)	Memory usage (bytes)	Inspected states (R1)	Inspected states (R2)	Inspected states (R3)
1	8.0005	13840	C	F,E,D	B,A,G
2	23.0013	13850	C	F,E,D	B,A,G
3	8.0005	13840	C	F,E,D	B,A,G
4	7.0004	13840	C	F,E,D	B,A,G
5	11.0006	13852	C	F,E,D	B,A,G
6	8.0004	13840	C	F,E,D	B,A,G
7	16.0009	13852	C	F,E,D	B,A,G
8	17.001	13840	C	F,E,D	B,A,G
9	17.001	14936	C	F,E,D	B,A,G
10	12.0006	13852	C	F,E,D	B,A,G

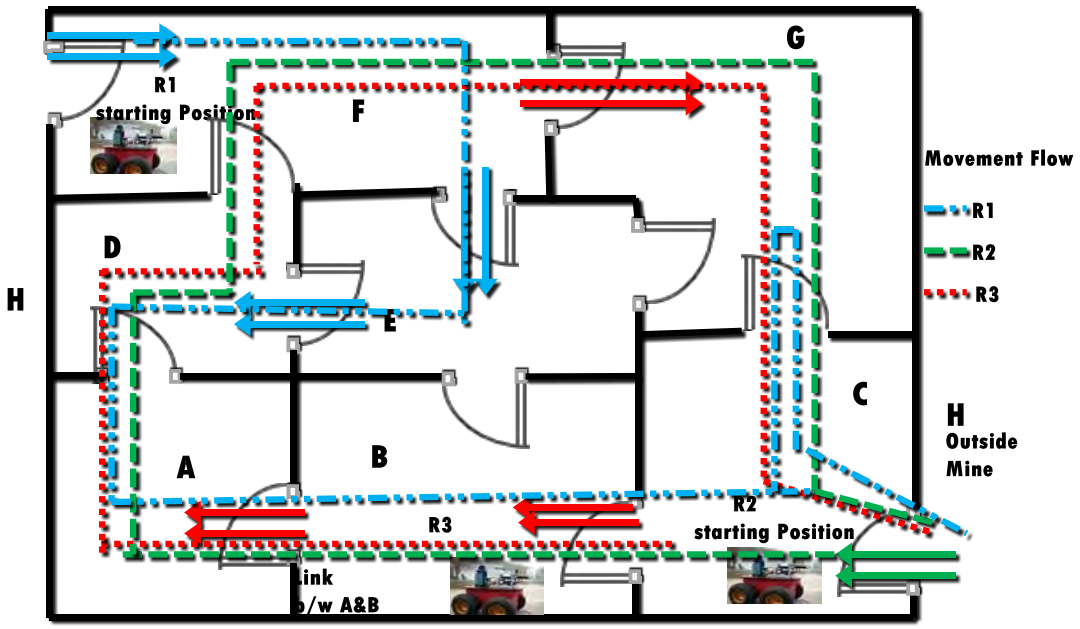


Figure 5.4: Movement of three robots showing cooperation

5.4 Experiment 3: Performance of QLACS Using Four Robots

To investigate the scalability of the QLACS model further, four autonomous robots' performance obtained by running the simulated model 10 times was analysed. It is interesting to note that the results show good cooperation between the four robots with an average time of 10.1 and memory usage range of 18,000 – 19,000. Though the memory increased slightly, more than for three robots, stable memory usage was maintained. The detail of the simulation result is laid out in Table 5.3. Figure 5.5 shows the simulated movement of the four robots. R1 and R3 enter the mine through state F and exit the mine through state C. The blue and purple arrows signify states inspected by R1 and R3 respectively. R2 and R4 enter the mine through state C and exit through state C. States inspected by R2 and R4 are shown by green and red double arrows respectively. The movement flow for R1 is FEDABCGC, for R2 is CBADFEGC, for R3 is FEDABCGC and for R4 is CBADFEGC.

Table 5.3: Performance of QLACS with four robots-based MRS

No of runs	Time (sec)	Memory usage (bytes)	Inspected states (R1)	Inspected states (R2)	Inspected states (R3)	Inspected states (R4)
1	10.0006	18320	F	C	G,E,D	B,A
2	14.0008	18320	F	C	G,E,D	B,A
3	10.0006	18320	F	C	G,E,D	B,A
4	11.0006	18868	F	C	G,E,D	B,A
5	8.0005	18312	F	C	G,E,D	B,A
6	10.0005	18320	F	C	G,E,D	B,A
7	11.0006	10320	F	C	G,E,D	B,A
8	10.0006	18320	F	C	G,E,D	B,A
9	7.0004	18200	F	C	G,E,D	B,A
10	10.0006	19408	F	C	G,E,D	B,A

A notable observation emanating from a comparison of Tables 5.2 and 5.3 is that there is a need for a larger mine area of inspection because R1 in Table 5.2 could inspect only state C, while R1 and R2 in Table 5.3 could inspect only F and C respectively. This implies that the size of the field of inspection is proportional to the number of robots to be deployed.

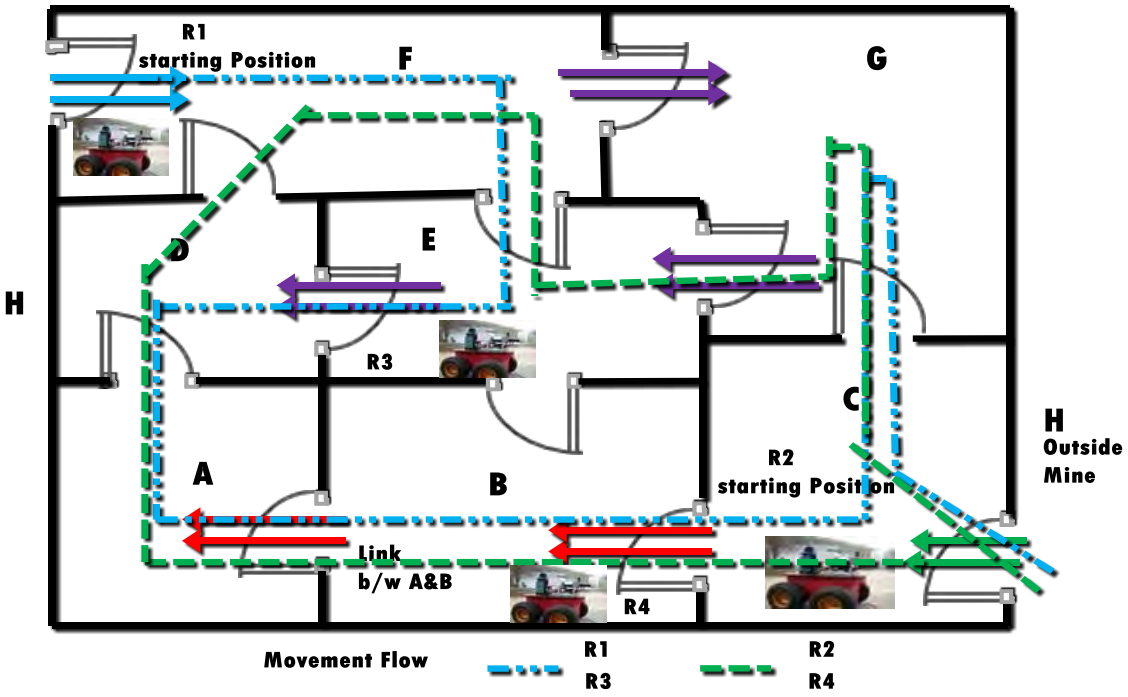


Figure 5.5: Movement of four robots showing cooperation

5.5 Experiment 4: Observations of Time and Memory Scalability

The major focus of this section is to test our proposed hybrid model for scalability performance. Figures 5.6 and 5.7 show moderate differences on robot results that attest to how scalable our

model is with respect to time, memory and number of robots. Looking at the trends of time and of memory for two, three, and four robots, the model promises to handle a reasonable number of robots even when the environment is expanded. For Figure 5.6, the blue curve depicts the trend of time for two robots using the QLACS model, the green curve depicts the trend for four robots using the same model and the red curve shows the trend for three robots under the same model. The trend with three robots is way off that of two and four robots; however, the model accommodated three robots. This goes to show that our double entry/exit has some effect on the result with odd and even number of robots. This is so because, in odd number of robots, one of the robots will be almost redundant because of double entry/exit points. The result on Figure 5.7 also shows a corresponding memory usage of two, three and four robots. As a proof of concept, one can deduce from the results in Figures 5.6 and 5.7 that QLACS is scalable in terms of the number of robots that can be used to achieve the inspection task under reasonable timing and memory usage.

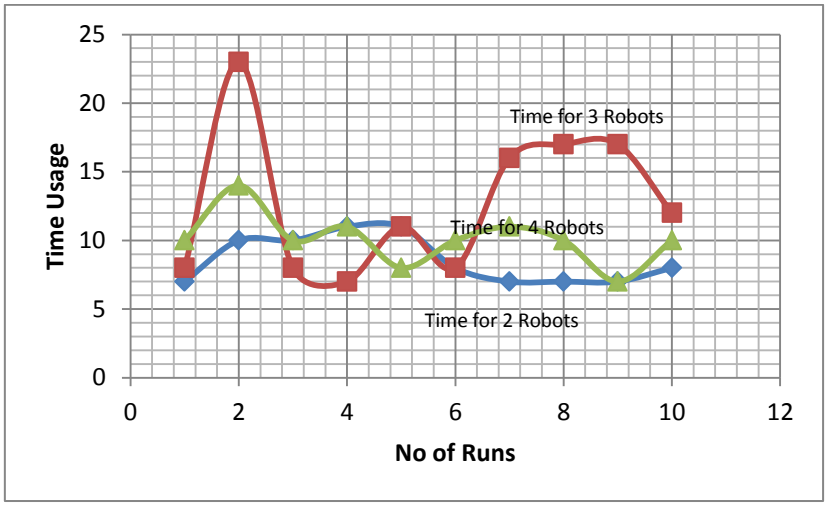


Figure 5.6: Trends of time with number of robots

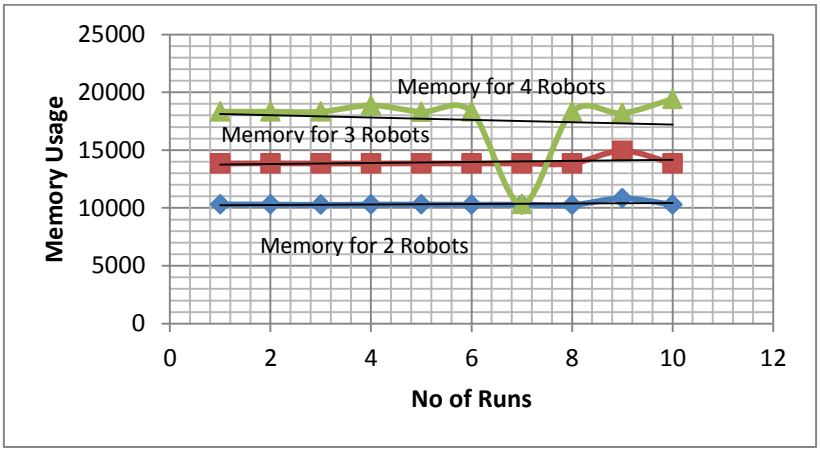


Figure 5.7: Trends of memory usage with number of robots

We also present some results obtained by experimenting with the cooperative behavioural action of two, three and four robots using the QLACS model. The performance of the two, three and four robots was evaluated by running the simulation three times, using the same number of agents. The performance of the proposed QLACS model shows good communication between the two, three and four robots under the states inspected, in the last four columns of Table 5.4. An almost stable time was used in achieving the inspection for all the robots. The detail of the simulation result is laid out in Table 5.4.

Table 5.4: Summary of scalability performance on QLACS

Row Numbers	**No of Robots**	**Time(sec)**	**Number of states inspected**			
			Robot 1 (R1)	**Robot 2 (R2)**	**Robot 3 (R3)**	**Robot 4 (R4)**
1	2	10.0006	4	3		
2	2	11.0007	4	3		
3	2	8.0005	4	3		
4	3	16.0009	1	3	3	
5	3	17.001	1	3	3	
6	3	12.0006	1	3	3	
7	4	11.0006	1	1	3	2
8	4	14.0006	1	1	3	2
9	4	10.006	1	1	3	2

A notable observation emanating from Table 5.4 is that there is a need for a larger mine area of inspection because R1 in rows 4 to 6 could inspect only one state, while R1 and R2 in rows 7 to 9 could inspect only one state each. This implies that the size of the field of inspection is proportional to the number of robots to be deployed.

5.6 Comparative Analysis of Cooperative Behaviour Systems

In investigating the similarities and differences of our cooperative behavioural model with other existing cooperative models, we use the criteria displayed in Table 5.5. Looking at the table, we can see from the features column that the four different models presented have different architectures, domains, intelligence, navigation approaches, conflict resolution methods and cooperation strategies.

However, not all four methods have a scalability feature. This thesis has shown how scalable our model is by experimenting with the performance of two, three and four robots.

Table 5.5: Comparative evaluation of existing cooperative behavioural system with our proposed system

S/N	Features	Action Selection Model[16]	Pheromone Route-finding Model[17]	Machine-learning Model[18]	QLACS Behavioural Model
1	**Architecture**	Learning and mediator modules	Knowledge-sharing through pheromones	Flexible two-layer multi-agent	Three-layer multi-robot
2	**Domain**	Robot soccer platform	Openstreetmap-based network layout	Object transportation task	Underground terrain safety inspection
3	**Scalability by Number of Robots**	Not scalable	Scalable	Not scalable	Scalable with time, memory and inspection
4	**Intelligence Methods**	Modular QL	Cooperative ACO algorithm	Integrated RL and GA	Integrated QL and ACS (QLACS)
5	**Navigation Approach**	Uni-vector field	Vehicle pheromones	RL & GA	Robots cooperative
6	**Conflict Resolution**	Coupled agent is proposed to resolve conflicts	Pheromone-related	Sequential QL algorithm	Broadcast and communication
7	**Cooperation Strategy**	Uni-vector field following	Route guidance through cooperative pheromones	Sequential QL	Information sharing

5.7 Discussions and Findings

This work has demonstrated the usefulness of enhancing the underground terrain pre-safety inspection with multi-robots as against humans. Investigation of the size of robots is used to demonstrate how scalable the proposed model is. The QLACS model is used as the intelligence that will help the robots to cope with the dynamic environment, find the optimal cooperation strategy and make the entire system flexible.

We conducted an experiment on different numbers of robots to prove the scalability of the system. The first experiment involved two robots using the QLACS model. The performance proved good communication and cooperation between them. We also conducted other experiments on the QLACS

model using three and four robots. They both revealed good communication and cooperation among the robots but needed an expansion of the mine environment to exhibit the full potential of the QLACS model in handling a different number of robots.

The efficiency of the cooperative behaviour is evaluated by a scaling relation between the task completion time, memory usage and the number of robots. The results as shown in Tables 5.1, 5.2 and 5.3 prove to enhance the cooperative behaviour of a team of robots. This will accelerate the rate of work output for MRS in the underground terrain.

The result of this work is an essential application that will reinforce multi-robot cooperative behaviours in any underground terrain task and in field robotics in general.

5.8 Chapter Summary

This chapter presents the quantitative evaluations of QLACS in terms of scalability measure. The comparative analysis of QLACS with other existing models was achieved. Discussions and findings of QLACS were also discussed in this chapter.

6. Conclusion and Future Work

6.1 Summary and Conclusion

This chapter concludes the thesis. A summary of contributions has shown how MRS behave cooperatively in underground terrains. The QL algorithm has been investigated for its potential quality of good communication, while the ACS algorithm was explored for good navigation properties. The strengths of the two algorithms have been harnessed and optimized to form a hybrid model QLACS, which addressed the behavioural situation in MRS effectively. QL is not a population-based algorithm but has been used for path-planning problems. The higher values in time and route costs for QL vindicate the inherent problem of it being chaotic and repetitive, i.e. a robot may repeat a state more often than necessary when it gets its path by QL. This wastefulness is due to the greedy search method used for the state selection. ACS, on the other hand, is a population-based algorithm where each ant's path is a potential solution. It has the advantage of delivering the best and optimal path among the solutions. QLACS leverages on this optimal path guaranteed by ACS to provide a global path for the agents to make an inspection decision by QL. The proposed model assumes an offline mode for the path planning algorithms, i.e. the global view of the map would have been provided before the learning robots start. For a global path view, a population-based algorithm will yield better results for both time and route costs. This is evident from the results in Tables 4.7 (page 76) and 4.8 (page 77) for seven states.

Comparing the performance of the two methods used in building our model in larger environments (i.e. increasing the number of nodes in the network), ACS will perform better in terms of optimized routes because several ants will deliver potential solutions, of which the best is handpicked for the inspection phase. The complexity and uncertainty of QL increase with larger environments because a robot could give any of the potential solutions in ACS, but one cannot guarantee it is the best possible path, unlike ACS, which will ensure the best possible one is delivered; thus QLACS will maintain an appreciable gain over QL for any number of nodes and robots. The results in Section 4.5 support this fact. The new hybrid model QLACS for cooperative behaviour of MRS in an underground terrain was proven able to find the optimal route and handle cooperation between two

robots effectively. The cooperative behavioural problem was narrowed down to two situations: navigational and cooperative behaviours. ACS was used to establish the shortest optimal route for two robots while the QL was used to achieve effective cooperation decisions. The results achieved after integrating the two algorithms showed a reduction in route costs and computation times, as shown in Figures 4.3 (page 76), 4.4 (page 77), 4.5 (page 78) and 4.6 (page 78) . The comparative analysis between QL, ACS and QLACS proved that the last-named is more robust and effective in achieving cooperative behaviour for MRS in the proposed domain.

6.2 Recommendation

Thorough experiments using MRS for underground terrain inspection have established the model's strengths and weaknesses. This work is recommended for underground terrains having the same representations as described in this thesis, i.e., using MRS to achieve an inspection task cooperatively where the local environment is unknown but the global view is known. The QLACS model achieved effective time for MRS navigation and inspection with optimized routes. The notion here is that MRS should cooperate while they navigate in the confined environment to expedite the inspection task and mitigate the risk involved in using humans to perform these and similar tasks. The solutions presented in this work will inspect any underground terrain within the bounds of described characteristics. Some of the parameters used in the model were motivated by the tasks of real-world applications. The model can be extended in a wider range. The inspection task was achieved with limited sensory information and without knowledge of the roof status and TG levels in the underground terrain. The capability and efficiency of the proposed model illustrated through simulations has been proven in this thesis. The model is applicable to some cooperative behaviour tasks in underground terrains.

6.3 Future Work

The result of this work will be used for future simulation and deployment of MRS applications in hazardous environments. An indication of expansion of this research area has conspicuously surfaced

in both real-life implementations and model increase. The model used in this work can also be improved upon by increasing the state space. For future work, more parameters, such as the size of the robot, the camera and its focal length, which could affect cooperative behaviour, should be considered when upgrading the proposed model. The results could also be improved if each thread of robot is attached to a processor on duo-core laptops, high performance computing or cluster computers. The relationship between the memory usage and the number of runs can be investigated when the number of runs is extended to higher digits of 15, 18, 20, etc.

Since the advent of cloud computing [100], several cloud simulators have been developed for performance. Simulation program for Elastic Cloud Infrastructures (SPECI) is a tool that allows analysing and exploration of scaling properties of large data center behaviour. For future study, cloud computing infrastructures will be explored for the simulations.

Appendix A

QLACS: A multi-robot cooperative behavioural model simulation environment.

A multi-robot cooperative simulator was developed in C#, using the dotnet development platform. This appendix briefly explains some important aspects of the simulation process. In each cycle, each robot makes a decision on where it wants to move next by communicating to the other robots.

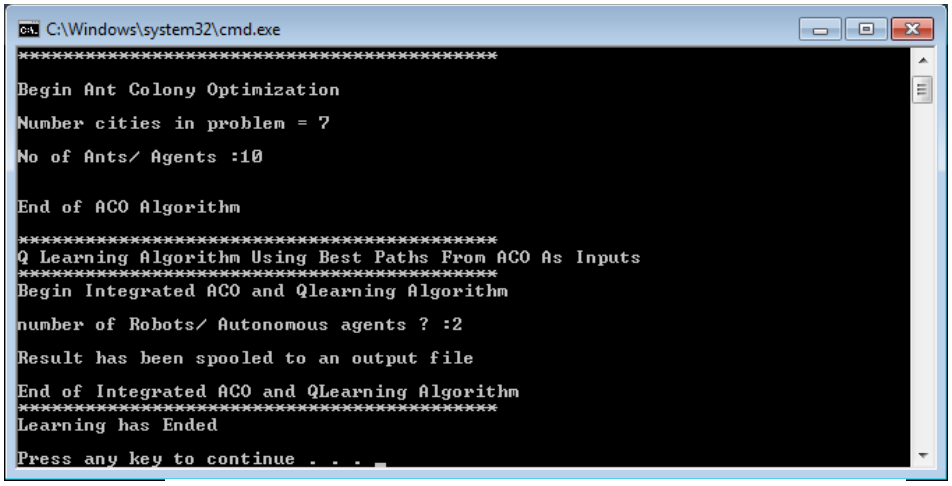
Route Finding

This is a key component of the algorithms proposed in this thesis. All robots in the team use simple ACS route finder which performs sufficiently well as shown in Figures A.1 and A.2.

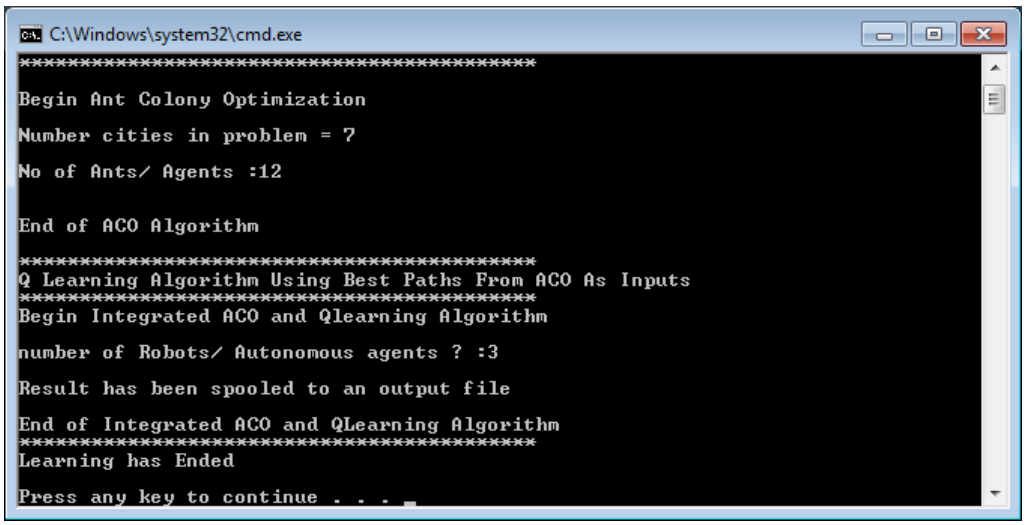
Communication

QLACS supports a good communication model achieved using QL. All robots are assumed to be within range and each robot communicates to another using the QL model (see chapter 3 for equations). Figures A.1 and A.2 show a combined route finder and communication model working together to achieve a cooperative task.

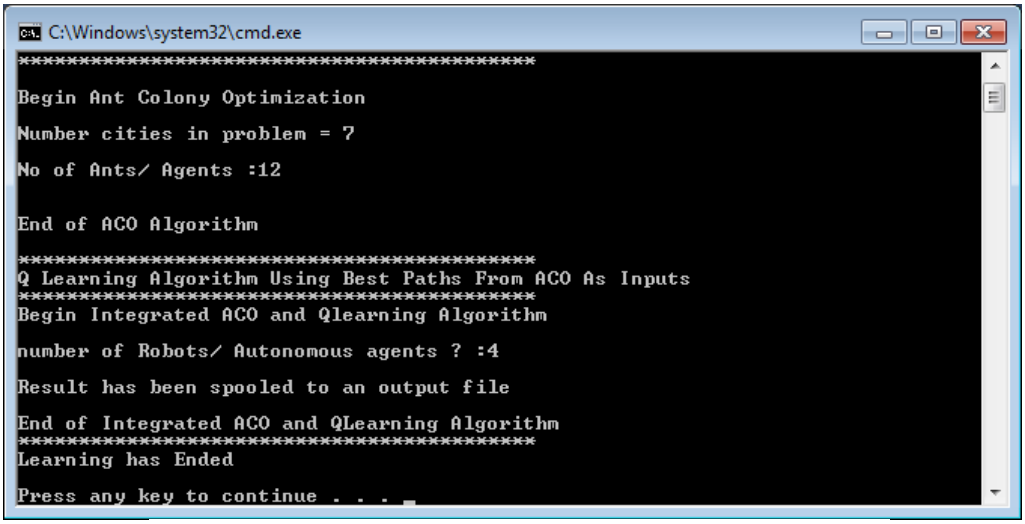
QLACS simulator is a quick, simple and controlled cooperative behavioural tool. The experiments conducted with this tool in this thesis confirmed this.



(a) Route finding and communication for 2 robots



(b) Route finding and communication for 3 robots



(c) Route finding and communication for 4 robots

Figure A.1: Demonstrating QLACS Simulation

List of References

[1] L.E. Parker: Current Research in Multi-robot Systems. In Seventh International Symposium on Artificial Life and Robotics (ISAROB), Vol. 7, pp.1 – 5, 2003.

[2] R. S. Aylett, D. P. Barnes: A Multi-robot Architecture for Planetary Rovers. In Proceeding of 5th ESA Workshop on Space Robotics, ASTRA, 1998.

[3] H. Yulan, Z. Qisong, X. Pengfei: Study on Multi-robot Cooperation Stalking Using Finite State Machine. In International Workshop on Information and Electronics Engineering (IWIEE), Vol. 29, pp. 3502 – 3506, 2012.

[4] M. Mataric, M. Nilsson, K.T. Simsarian: Cooperative multi-robot box-pushing. In Proceeding of IEEE/RSJ IROS, Vol. 3, pp. 556 – 561, 1995.

[5] D. Rus, B. Donald, J. Jennings: Moving Furniture with Teams of Autonomous Robots. In Proceeding of IEEE/RSJ IROS, Vol. 1, pp. 235 – 242, 1995.

[6] L.E. Parker, C. Touzet: Multi-robot Learning in a Cooperative Observation Task. Distributed Autonomous Robotic Systems 4, ISBN: 978-4-431-67919-6, pp. 391 – 401, 2000.

[7] F. W. Heger, L. M. Hiatt, B. Sellner, R. Simmons, S. Singh: Results in Sliding Autonomy for Multi-Robot Spatial Assembly. In 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space, pp. 5–8, 2005.

[8] L. E. Parker: ALLIANCE: An Architecture for Fault-tolerant Multi-robot Cooperation. IEEE Trans. on Robotics. and Automation, Vol. 14, pp. 220 -240, 1998.

[9] P. Stone, M. Sridharan, D. Stronger, G. Kuhlmann, N. Kohl, P. Fidelman, N.K. Jong: From Pixels to Multi-Robot Decision-Making: A Study in Uncertainty. Journal of Robotics and Autonomous Systems, Vol. 54, pp. 933-943, 2006.

[10] J. Fink, N. Michael, S. Kim, V. Kumar: Planning and Control for Cooperative Manipulation and Transportation with Aerial Robots. In International Symposium of Robotics Research 2009.

[11] E. Yang, D. Gu: Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey. In Proceeding of IEEE symposium on Computational Intelligence (CIG) and Games, pp. 292 – 299, 2005.

[12] T. Taipale, S. Hirai: A Behaviour-Based Control System Applied over Multi-Robot System. In Proceedings of the IEEE/RSJ International Conference on Robots and Systems, vol. 3, pp 1941 – 1943, 1993.

[13] W.A. Groves, V.J. Kecojevic, D. Komljenovic: Analysis of Fatalities and Injuries involving Mining Equipment. Journal of Safety Research, Vol. 38, pp. 461 – 47, 2007.

[14] H. L. Hartman and J. M. Mutmansky: Introductory Mining Engineering, 2nd ed. John Wiley & Sons Inc., ISBN: 0471348511, Oct. 2002.

[15] S. Yarkan, S. Guzelgoz, H. Arslan, R.R. Murphy: Underground mine Communications: A Survey. IEEE Communication Surveys & Tutorials, vol. 11, no. 3, and pp. 125 – 142, 2009.

[16] J.P. Leger Trends and Causes of Fatalities in South African mines. Journal of Safety Science, vol. 14, pp. 169 – 185, 1991.

[17] D. Dudek, M. Jenkin, E. Milios, D. Wilkes: A Taxonomy for Multi-agent Robotics. Journal of Autonomous Robots, Vol. 3, pp. 375 – 397, 1996.

[18] R.M. Murray: Recent Research in Cooperative Control of Multivehicle Systems, Trans. ASME Journal of Dynamic System, Measurement Control, Vol. 129, no 5, pp. 571 – 583, 2007.

[19] A. Farinelli, L. Iocchi, and D. Nardi: Multirobot systems: A classification focused on coordination. IEEE Trans. Syst., Man, Cybern., Part B: Cybern., vol. 34, no. 5, pp. 2015–2028, 2004.

[20] K. Tanaka and E. Kondo: A scalable localization algorithm for high dimensional features and multirobot systems. In Proceeding of IEEE International Conference on Network, Sensor and Control, pp. 920–925, 2008.

[21] S. X. Yang and M. Meng: Neural network approaches to dynamic collision-free trajectory generation. IEEE Trans. Syst., Man, Cybern., Part B: Cybern., vol. 31, no. 3, pp. 302–318, 2001.

[22] D. Portugal, R.P. Rocha: On the Performance and Scalability of Multi-Robot Patrolling Algorithms. In proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics, pp. 50 – 55, 2011.

[23] T. Yasuda, K. Ohkura, K..A. Ueda: A Homogeneous Mobile Robot Team that is Fault-Tolerant. In Journal of Advanced Engineering Informatics , Vol. 20, pp.301-311, 2006.

[24] Y.B. Wu: Blood Coal: A Flawed Development Strategy behind the Death Toll in Chinese Coal Mines, Jackson School of International Studies, University of Washington, 2008.

[25] Chamber of Mines of South Africa. Fact & Figures 2013. Available online: http://www.bullion.org.za/content/?pid=71&pagename=Facts+and+Figures (accessed on 13 April, 2014).

[26] M. Teke: This is the Mining Industry. Available online: http://www.sacollierymanagers.org.za/docs/South%20African%20Mining%20industry%20Mike%20Teke%204%20October%202013.pdf (accessed on 04 April, 2014).

[27] S. Koenig, R.G. Simmons: Complexity Analysis of Real-Time Reinforcement Learning, In proceedings of the AAAI conference on Artificial Intelligence, pp. 99 – 105, 1993.

[28] World's Worst Mining Disasters. http://worldnews.about.com/od/disasters/tp/Worlds-Worst-Mining-Disasters.htm. (accessed on 04 April 2014).

[29] F. Schulz, D. Wagner, K. Weihe: Dijkstra's algorithm online: An empirical case study from public railroad transport. Journal of Experimental Algorithmics, 5(12), 2000.

[30] F. Neumann, D. Sudholt, C.Witt: Computational Complexity of Ant Colony Optimization and its Hybridization with Local Search. Innovations in Swarm Intelligence, pp. 91 – 120, 2009.

[31] Automatic Robotic Vehicle: http://theinstitute.ieee.org/career-and-education/preuniversity-education/autonomous-robotic-vehicle-earnsstudent-ieee-scholarship (accessed on 04 April 2014).

[32] J.H. Saleh, A.M. Cummings: Safety in the Mining Industry and the Unfinished Legacy of Mining accidents: Safety Levers and Defense-in-depth for Addressing Mining Hazards. Journal of Safety Science, Vol. 49, pp. 764 – 777, 2011.

[33] J.P. Leger: Trends and causes of Fatalities in South African Mines. Journal of Safety Science, Vol. 14, PP. 169 – 185, 1991.

[34] D.F. Tver, R.W. Bolz: Robotics Source Book and Dictionary, Industrial Press, New York, 1983.

[35] B.S. Dhilion: Robot Safety Analysis Methods. Department of Mechanical Eng. University of Ottawa, Canada, 2003.

[36] D. Laurence: Safety Rules and Regulations on Mine Sites – The Problem and a Solution. Journal of Safety Research, Vol. 36, pp. 39 – 50, 2005.

[37] Occupational Safety and Health: http://www.anglogold.com/subwebs/informationforinvestors/reporttosociety05/values_bus_principles/safety_health/sh_cs_sa_5_9.htm (accessed on 07 May 2014).

[38] S. Molina, I. Soto, R. Carrasco: Detection of Gases and Collapses in Underground Mines using WSN. In Industrial Technology (ICIT) and IEEE International Conference, pp. 212 – 225, 2011.

[39] D.P. Stormont, V.H. Allan: Managing Risk in Disaster Scenarios with Autonomous Robots. In Journal of Systemics, Cybernatics and Informatics (IIISCI), Vol. 7, pp. 66 – 71, 2009.

[40] Mine Health and Safety Council Annual Report 2009-2010, 2003-2004.

[41] G. Wei: Statistical Analysis of Sino-U.S. Coal Mining Industry Accidents. International Journal of Business Administration, vol. 2, pp. 82 – 86, 2011.

[42] J. A. Leitner: Survey of Multi-Robot Cooperation in Space. AT-EQUAL ECSIS Symposium on Learning and Adaptive Behavior in Robotics Systems, Vol. 2, pp.144 – 151, 2009.

[43] C. Cai, C. Yang, Q. Zhu, Y. Liang: Collision Avoidance in Multi-Robot Systems. In Proceedings of the IEEE International Conference on Mechatronics & Automation, 2007.

[44] E. Hanna, P. Straznicky, R. Goubran: Obstacle Detection for Low Flying Unmanned Aerial Vehicles using Stereoscopic Imaging. In IEEE International Instrumentation and Measurement Technology Conference Victoria, 2008.

[45] A. Rosenfeld, G.A.; Kaminka, S. A.; Kraus: Study of Scalability Properties in Robotic Teams. Bar Ilan University, Ramat Gan, Israel.

[46] P. Kui-Hong, K. Yong-Jae, K. Jong-Hwan: Modular Q-Learning Based Multi-Agent Cooperation for Robot Soccer. Journal of Robotics and Autonomous Systems, Vol. 35, pp. 109 – 122, 2001.

[47] A. Sadegh: Machine Learning in a Multi-Robot System. Department de Electique de l'Ecole Polytechnique de Montreal, Canada.

[48] G. Beni, J. Wang: Swarm Intelligence in Cellular Robotic Systems. In Proc. NATO Advanced Workshop on Robotics and Biological Systems, 1989.

[49] A. Runka: Evolving an edge Selection Formula for Ant Colony Optimization. Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1075 – 1082, 2009.

[50] W. Duch: What is Computational Intelligence and Where is it Going?. Department of Informatics, Nicolaus Copernicus University, Poland.

[51] L. Dawson, I.Stewart: Improving Ant Colony Optimization performance on the GPU using CUDA, IEEE congress on Evolutionary Computation, pp. 1901 – 1908, 2013.

[52] E. Bonabeau, M. Dorigo, G. Theraulaz: Swarm Intelligence, from Natural to Artificial Systems, Oxford University Press, 1999.

[53] Y. Liu, K.M. Passino: Swarm Intelligence: Literature Overview, Department of Electrical Engineering, University of Ohio, 2000.

[54] L. Li, A. Martinoli, Y.S. Abu-Mostafa: Emergent Specialization in Swarm Systems, IDEAL, LNCS 2421, pp. 261 – 266, 2002.

[55] B. Englot, F. Hover: Multi-Goal Feasible Path Planning Using Ant Colony Optimization, IEEE International Conference on Robotics and Automation, pp. 2255 – 2260, 2011.

[56] Z.N. Naqiv, H. Matheru, K. Chadha: Review of Ant colony Optimization on Vehicle Routing Problems and Introduction to Estimated-Based ACO, International Conference on Environment Science and Engineering IPCBEE, Vol. 8, pp. 161 – 166, 2011.

[57] M. Dorigo, T. St¨utzle: The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, Technical Report IRIDIA, 2000.

[58] R. Claes, T. Holvoe: Cooperative Ant Colony Optimization in Traffic Route Calculations. Advances in Intelligent and Soft Computing, vol. 155, pp. 23 – 34, 2012.

[59] A. Chella, G. Lo Re, I. Macaluso, M. Ortolani, D. Peri: A Networking Framework for Multi-Robot System, Recent Advances in Multi-Robot Systems, pp. 1 - 14, 2008.

[60] M. Peasgood, J. McPhee, C. Clark: Complete and Scalable Multi-Robot Planning in Tunnel Environments, Digital commons, 2006.

[61] R. Zlot, A. Stenz: Market-Based Multi-Robot Coordination for Complex Task. International Journal of Robotics Research, vol. 25, pp. 1- 26, 2006.

[62] F.E. Schneider, D. Wildermuth, M. Moors: Methods and Experiments for Hazardous area activities using Multi-Robot System. IEEE international Conference on Robotics and Application, pp. 3559-3564, 2004.

[63] C. Thorp, H. Durrant-Whyte: Field Robots. In Tenth International Symposium on Robotics Research, Vol. 6, pp. 329 – 240, 2003.

[64] S.R. Teleka, J.J. Green, S. Brink, J. Sheer, K. Hlophe: The Automation of the 'Making Safe' Process in South African Hard-Rock Underground Mines. International Journal of Engineering and Advanced Technology (IJEAT) Vol. 1, pp. 1-7, 2012.

[65] C.O. Yinka-Banjo, I.O. Osunmakinde, A. Bagula: Autonomous Multi-robot Behaviours for Safety Inspection under the Constraints of Underground Mine Terrains. Ubiquitous Computing and Communication Journal; ISSN 1992-8424, (7) 5,pp 1316 – 1328, 2012.

[66] M.J. Mataric: Reinforcement Learning in the Multi-robot Domain. Autonomous Robots, Vol. 4, pp. 73 – 83, 1997.

[67] M.M. Botvinick, Y. Niv, A.C. Barto: Hierarchically Organized Behaviour and its Neural Foundations: A Reinforcement Learning Perspective: Cognition, Vol. 113, pp. 262 – 280, 2009.

[68] M. Dorigo, L.M. Gambardella: Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics , 1996.

[69] E.S. Redden, R.A. Pettitt, C.B. Carstens, L.R. Elliott: Scalability of Robotic Displays: Display Size Investigation. In Army Research Laboratory, 2008.

[70] J. Mulder, X. Wang, F. Ferwerda, M., Cao: Mobile Sensor Networks for Inspection Tasks in Harsh Industrial Environments. Sensors, vol. 10, pp. 1599 – 1618, 2010.

[71] J. Ni, S.X. Yang: Bioinspired Neural Network for Real-Time Cooperative Hunting by Multi-robots in Unknown Environments. IEEE Transactions on Neural Networks, Vol. 22, pp. 2062 – 2077, 2011.

[72] J. Ni, S.X. Yang: A F uzzy-Logic Bsed Chaos GA for Cooperative Foraging of Multi-Robots in Unknown Environments. International Journal of Robotics and Automation, vol. 27, pp. 15 – 30, 2012.

[73] Y. Wang, C.W. de Silva: A Machine-Learning Approach to Multi-Robot Coordination. Engineering Applications of Artificial Intelligence, vol. 21, pp. 470 – 484, 2008.

[74] G. Lee, N.Y. Chong: Decentralized Formation Control for Small-Scale Robot Teams with Anonymity. Mechatronics, vol. 19, pp. 85 – 105, 2009.

[75] T. Guan-Zheng, H. Huan, S. Aaron: Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots. Acta Autmatica Sinica, Vol. 33, pp. 279 – 285, 2007.

[76] M. Ahmed, M.R. Khan, M.M. Billah, S. Farhana: A Novel Navigation Algorithm for Collaborative Multi Robots, European Journal of Scientific Research, Vol. 41, pp. 472 – 481, 2010.

[77] M.A. Rahman, M. S. Miah, W. Gueaieb, A. El Saddik: Senora: A P2P Service Oriented for Collaborative Multi-robot Sensor Networks. IEEE Sensors Journal, pp. 1 – 9, 2007

[78] P. Nebot, J. Torres-Sospedra, R.J. Martinez: A New HLA-Based Distributed Control Architecture for Agricultural Teams of Robots in Hybrid Applications with Real and Simulated Devices or Environments. Sensors, vol. 11, pp. 4385 – 4400, 2011.

[79] J.N. Bakambu: Integrated Autonomous System for Exploration and Navigation in Underground Mines. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2308 – 2313, 2006.

[80] Z. Shaogang, L. Ming: Path Planning of Inspection Robot Based on Ant Colony Optimization Algorithm. IEEE Computer Society, pp. 1474 – 1477, 2010.

[81] K.M. Slm, W.H. Sun: Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions. IEEE Transactions on Systems. Man. and Cybernetics, vol. 33, pp. 560 – 572, 2003.

[82] L.M. Gambardella, M., Dorigo: Ant-Q: A Reinforcement Learning Approach to Travelling Salesman Problem. 12th International Conference on Machine Learning, pp. 252 – 260, 1995.

[83] X. Deng, L. Zhang, L. Luo: An Improved Ant Colony Optimization Applied in Robot Path Planning Problem. Journal of Computers, vol. 8, pp. 585 – 593, 2013.

[84] N. Buniyamin, N. Sariff, W.A.J . Wan Ngah, Z. Mohamad: Robot Global Planning Overview and a Variation of Ant Colony System Algorithm. International Journal of Mathematical and Computers in Simulation, Vol. 5, pp. 5 – 16, 2011.

[85] J. Chia-Feng, L. Chun-Ming: Ant Colony Optimization Incorporated with Fuzzy Q-Learning for Reinforcement Fuzzy Control. IEEE Transactions on Systems, Man. And Cybernetics, vol. 39, pp. 597 – 608, 2009.

[86] R.A.C. Bianchi, C.H.C. Ribeiro, A.H.R. Costa: On the Relation between Ant Colony Optimization and Heuristically Accelerated Reinforcement Learning. International workshop on Hybrid Control of Autonomous Systems, pp. 49 – 55, 2009.

[87] E.J. Dries, G.L. Peterson: Scaling Ant Colony Optimization with Hierarchical Reinforcement Learning Partitioning. Association for Computing Machinery (ACM), pp. 25 – 32, 2008.

[88] L. Ondrej, M. Milos: Fuzzy Manual Control of Multi-Robot System with Built-in Swarm Behaviour. Human System Interaction, HSI IEEE, pp. 4 – 9, 2009.

[89] T.L. Anderson, M. Donath: Animal Behaviour as a Paradigm for Developing Robot Autonomy, Robotics and Autonomous Systems, vol. 6 pp. 145 – 168, 1990.

[90] M. Kudelski, M. Cinus, L. Gambardella, G.A. Di Caro: A Framework for Realistic Simulation of Networked Multi-Robot Systems. International Conference on Intelligent Robots and Systems (IROS), pp. 5018 – 5025, 2012.

[91] K. Fregene, D. Kennedy, D. Wang: Multi-Vehicle Pursuit-Evasion: an Agent-Based Framework. Proceedings of IEEE International Conference on Robotics and Automation, pp. 2707 – 2713, 2003.

[92] E.D.S. Costa, M.M. Gouvea Jr: Autonomous Navigation in Dynamic Environments with Reinforcement Learning and heuristic. IEEE International Conference on Machine Learning and Applications, pp. 37 – 42, 2010.

[93] H. Yamaguchi: A Distributed Motion Coordination strategy for Multiple nonholonomic Mobile Robots in Cooperative Hunting Operations. Robotics and Antonomous Systems, vol. 43, pp. 257 – 282, 2003.

[94] H.W. Stone, G. Edmonds: A Hazardous Materials Emergency Response Mobile Robot. In Proceedings of the IEEE International conference on Robotics and Automation (ICRA), pp. 67-73, vol.1, 1992.

[95] M.J. Mataric: Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents. In Journal of Cognitive Systems Research 2, pp. 81-93, 2001.

[96] M.B. Diaz, A. Stentz: A Free Market Architecture for Distributed Control of a Multirobot System. In 6th International Conference on Intelligent Autonomous Systems (IAS-6), pp. 115-122, July, 2000.

[97] P. Anawat, R. Rolf, V. Juris, R. David: Market-Based Co-Evolution planning for Multiple Autonomous Vehicles. American Institute of Aeronautics and Astronautics, University of Washington, USA.

[98] J. Mclurkin, D. Yamins: Dynamic Task Assignment in Robot Swarms. In Proceedings of Robotics, Science and Systems, 2005.

[99] I. Susnea, M. Viorel, V. Grigore: Simple, Real-time Obstacle avoidance algorithm for mobile robots. In 8th WSEAS International Conference on Computational Intelligence, Man-machine Systems and Cybernetics (CIMMACS), 2009.

[100] R. Malhotra, P. Jain: Study and Comparison of Various Cloud Simulators Available in the Cloud Computing. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Issue 9, pp. 347 – 350, 2013.

[101] F. Neumann, C. Witt: Runtime analysis of a simple ant colony optimization algorithm. Algorithmica, Vol. 54, Issue 2, pp. 243-255, 2009.

[102] M. Wooldridge: An Introduction to MultiAgent Systems. Wiley Publishing, 2009.